

An Improved MAD-MARS Beam Line Builder: User's Guide*

M.A. Kostin, O.E. Krivosheev[†], N.V. Mokhov and I.S. Tropin[‡]

Fermi National Accelerator Laboratory, P.O. Box 500, Batavia, Illinois 60510

[†] *Elekta/Precision Therapy Inc., Norcross, GA 30071*

[‡] *Tomsk Polytechnic University, Tomsk, 634034, Russia*

September 8, 2004

Abstract

The MAD-MARS Beam Line Builder (MMBLB) is a tool that helps the MARS14 and MARS15 Monte Carlo codes build models of beam lines and accelerators from MAD OPTICS files. A substantially revised version of the Beam Line Builder has been developed. This paper comprises a short User's Guide to the new MMBLB.

*This work was supported by the Universities Research Association, Inc., under contract DE-AC02-76CH03000 with the U.S. Department of Energy.

1 Introduction

Monte Carlo simulations of beam-induced energy deposition effects in accelerators and beam lines is one of the main applications of the MARS code [1, 2]. Other aspects of beam line and accelerator design are addressed with other codes that implement lattice calculations and efficiently track particles through the structure. The use of multiple codes for the design can lead to the problems of model inconsistencies, performance penalties and increased time needed for developing the models. The MAD-MARS Beam Line Builder, MMBLB, is a code that mitigates these problems by using a beam line description employed by accelerator design codes such as MAD [3] to create a suitable input to MARS. An earlier version of MMBLB was described in Ref. [4, 5]. It has been successfully used in a number of applications for the Fermilab Booster [6, 7], Tevatron, Proton Driver [8, 9], NuMI beam line [10, 11] and Japanese J-PARC [12] to study beam loss distributions and radiation fields and to design beam collimation systems.

The purpose of this paper is to introduce a significantly improved version of MMBLB and provide a User's Guide to that.

2 New Features

This revised version of MMBLB incorporates significant new features and enhancements. Some of the changes are designed to minimize potential incompatibilities with lattice descriptions created with MAD and other similar codes. Others expand the capabilities of MARS. These new features are:

- The set of supported element types has been extended to encompass almost all elements supported by MAD (see Table 1 in Section 5 below).
- An arbitrary number of beam lines can be put in a model.
- The beam lines can be 3-D. In the older version, only planar beam lines were allowed.
- The beginning of a beam line can be placed in an arbitrary point of space with an arbitrary orientation.
- More sophisticated algorithms and new data structures enable more efficient searches through the beam line geometry.

3 Overview of MMBLB Functionality

MMBLB is a library of specialized functions that is included in the standard MARS distribution package. The library serves to process the geometry of beam line elements within the MARS framework. MMBLB is activated when the 13th index in the MARS input file MARS.INP is set to "T" (INDX 13=T).

A MMBLB process has phases of initialization and calculation. During initialization, the MARS subroutine REG1 calls the customizable subroutine BLINIT, in which a problem specific user-written initialization code is included. The initialization code normally reserves the computer memory for the requested number of beam lines (subroutine BEAMLINES_IN_USE); reads the positions of elements from MAD OPTICS files (format of the OPTICS files is described in Section 4) and puts them into a MARS global frame in accordance with the requested position and orientation of the first elements of the beam lines (BUILDBL); sets parameters for the reference orbits (SETEBL); collects the names of sub-zones in the beam lines (BLNAME) and their material indexes (BLMAT). The volumes of sub-zones are collected by the subroutine BLVOL which is invoked from VFAN. The subroutines BLNAME, BLMAT and BLVOL look for the requested information in internal data structures of MMBLB by sending inquires to each element in the beam lines. The inquires are processed then by designated functions provided for each type of beam elements (Sections 6, 8.3). In order to bind these functions with the elements, they must be registered first by calling MARS_EL_REGISTER. The registration must be provided by the user in the subroutine BLGEOINIT, which is called by MMBLB at the initialization stage. The subroutine MARS_EL_REGISTER also takes other functions as arguments. The functions initialize data structures specific for a given element and describe its geometry and fields.

The user has an option to print the beam line contents into a file with the subroutine BLPRINT after a successful initialization. The file helps interpret the MARS output. The file is self-explanatory. The fields in the file are the type of element, its name, position in the beam line (s -coordinate), length, and the first and last non-standard MARS zones assigned to the element.

During the calculation stage, MMBLB finds the zone number and corresponding material index for a point with the given coordinates. The user invokes the subroutine BLGEO from REG1 in order to do that. The magnetic field at the point is calculated by BLFIELD, the subroutine called from FIELD. The user must specify the beam line index to be used by calling SET_CURRENT_BEAMLINE, since MMBLB can only work with one beam line at a time.

MMBLB provides a relatively simple method to describe beam enclosures (Section 7). It can be done in a variety of ways. The geometry of the enclosures is described by the user in the subroutine TUNNEL_GEO, which is automatically called by MMBLB. The subroutine is especially appropriate for enclosures with cross-sections remaining same within long stretches, and when the enclosures follow the beam lines. The materials, names and volumes for zones of the beam enclosure have to be properly initialized.

MMBLB offers a set of subroutines that allows the user to transform the coordinates from the MARS frame to a local one of a beam line and vice versa. The related subroutines are FLAT2GLOB, GLOB2FLAT, MAD2MARS, MARS2MAD, BML2MARS and MARS2BML. The subroutines BML2MARS and MARS2BML are customizable and should normally use FLAT2GLOB, GLOB2FLAT, MAD2MARS and MARS2MAD. This functionality is very useful for collecting histograms in a local frame of a beam line instead of the MARS global frame.

4 Rules for Building Beam Lines

An input for the MMBLB system is a MAD OPTICS file [3]. An OPTICS file is a plain text file that describes longitudinal positions of elements along the beam line, element orientation and field parameters. An example of an OPTICS file is given below. The first line of the example describes the notations used and is not a part of the OPTICS file.

KEYWORD	TYPE	NAME	S(m)	L(m)	K0L	K1L	K2L	K3L	TILT
"LINE"	"~"	"NUMI"	0	0	0		0	0	0
"MARKER"	"~"	"LBEG"	0	0	0		0	0	0
"KICKER"	"MILAM"	"BLAM1"	3.048	3.048	0		0	0	0
"DRIFT"	"DRIFTNUMI"	"WMAD"	3.3528	0.3048	0		0	0	0
"QUADRUPOLE"	"MIQUAD"	"HQ608"	6.4008	3.0480	0		0	0	0
"DRIFT"	"DRIFTNUMI"	"DS2"	6.7056	0.3048	0		0	0	0
"RBEND"	"EPB"	"MC5U1"	9.7536	3.0480	-0.76E-2	0	0	0	1.5708
"DRIFT"	"DRIFTNUMI"	"DS"	10.0584	0.3048	0		0	0	0

An element is described with a single line. The first three fields are the MAD keyword, type and name of the element. Those serve for unambiguous identification of an element with a unique structure. The keyword can not be shortened, although this is a normal practice in MAD. That means that the keyword, *e.g.* "KICK", can not be used in lieu of "KICKER". There are two methods to identify elements: 1) using the exact match of the type and name, 2) if this fails, then only the type is used to find the element. The details on how to use these methods in practice are given below. Note that if two similar elements differ only by their lengths, and hence the volumes of zones, they should be declared under different types. That may not apply to a situation when the volumes of zones are not of interest.

The next two fields in the OPTICS file are the s -position of the end of the element and the element length L expressed in meters. The length of straight elements and magnets of the type "RBEND" is treated by MMBLB as the distance between the points where a reference orbit enters and exits the elements. For sector magnets, denoted with the keyword "SBEND", the length is that of a part of a reference orbit that pertains to the element (see Fig. 1).

The next four parameters $K0L$, $K1L$, $K2L$ and $K3L$ are related to the central magnetic field. The meaning of them is the magnet "strength" in dipole, quadrupole, sextupole and octupole components of the B-field multiplied by the magnet length. The analytical expression for the strength is this:

$$K_n = \frac{\partial^n \vec{B}}{\partial \vec{x}^n} \frac{1}{B_\rho} = \frac{\partial^n \vec{B}}{\partial \vec{x}^n} \frac{e}{p_0}, \quad (1)$$

where $B_\rho = p_0/e$ is the magnet rigidity, e is the charge of accelerated particle and p_0 is the momentum for a particle on the reference orbit. This way, the parameter $K0L$ becomes the

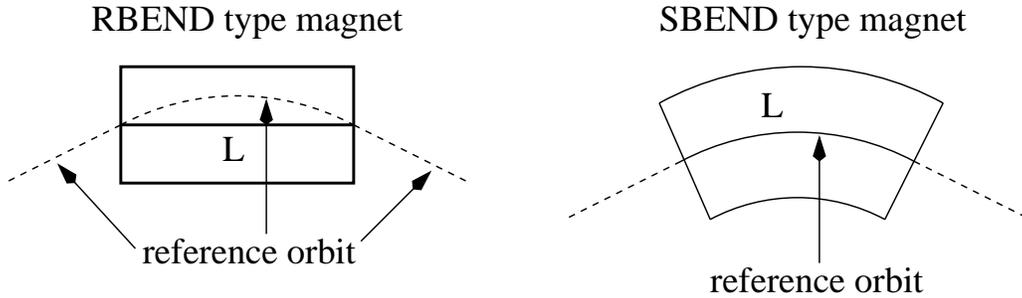


Figure 1: Definition for the element length L for RBEND and SBEND dipoles.

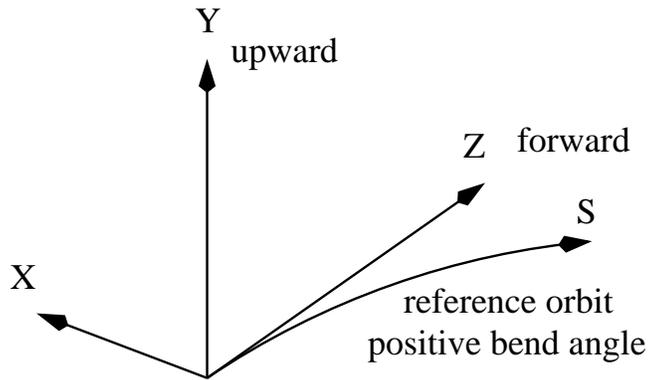


Figure 2: Definition for the MAD frame.

bend angle of a dipole. In accordance with the MAD rules, a positive bend angle represents a bend to the right, i.e. towards negative x -values (Fig. 2).

The last parameter is the roll angle of the element around the s -axis. A positive angle forms a right-hand screw with the s -axis. The roll angle makes it possible to define a 3-D beam line with practically any orientation of its elements. Position and orientation of the elements are entirely governed by the rules of MAD [3].

There are no restrictions on the number of beam lines in a MARS model in this version of MMBLB. Also, a beam line can be placed in an arbitrary point of the MARS global frame. The orientation of the first element of the beam line can also be specified by the user (subroutine `BUILDBL`).

5 What MMBLB Does and Does not Support

The elements, specified with the keywords summarized in Table 1, are supported by the MMBLB system and their parameters are defined in an `OPTICS` file. If, for whatever reason, the length of an element is zero, but its length must be greater than that as given in Table 1,

Table 1: List of the MAD keywords supported by MMBLB for elements whose parameters defined in OPTICS file.

Keyword	Specific traits
LINE	no effect on beam line, $L = 0$
MARKER	no effect on beam line, $L = 0$
DRIFT	drift space, $L > 0$
SBEND	sector bending magnet, $L > 0$
RBEND	rectangular bending magnet, $L > 0$
QUADRUPOLE	quadrupole, $L > 0$
SEXTUPOLE	sextupole, $L > 0$
OCTUPOLE	octupole, $L > 0$
MULTIPOLE	general thin multipole, $L = 0$
HMONITOR	orbit position monitor in horizontal plane, $L > 0$
VMONITOR	orbit position monitor in vertical plane, $L > 0$
MONITOR	orbit position monitor in both planes, $L > 0$
INSTRUMENT	space for beam instrumentation, $L > 0$
YROT	rotation around the y-axis, $L = 0$
SROT	rotation around the s-axis, $L = 0$

then this element is ignored.

Some keywords in Table 1 deserve a special consideration. The keywords “SROT” and “YROT” introduce an additional rotation of a reference orbit around the s - and y -axes, respectively, and do not describe an actual element. A multipole is normally used in MAD to take into account various field corrections of higher orders. Such corrections must be introduced by the MARS user in appropriate field functions. For example, the best proven practice is to use field maps for the magnets. The field maps are normally produced by the POISSON or OPERA finite element codes [13, 14]. The field maps contain all the possible corrections, so that the multipoles are ignored by MMBLB. The only exception is a multipole with the dipole component. Such a multipole has the same effect as a dipole of a zero length and the deflection angle of $K0L$.

The elements represented with the keywords “SOLENOID”, “HKICKER”, “VKICKER”, “KICKER”, “RFCAVITY”, “ELSEPARATOR”, “ECOLLIMATOR” and “RCOLLIMATOR” are also supported by MMBLB, but their parameters can not be provided in an OPTICS file. Instead, the parameters must be defined internally in the code. The same rule applies here: all the elements with zero lengths are ignored. Finally, the keywords “BEAMBEAM”, “MATRIX” and “LUMP” are not supported.

6 How to Define an Element

Whereas an OPTICS file controls the longitudinal position of a beam line element, the user must provide several subroutines that describe the element geometry, field, materials, volumes and names of its sub-zones. In order to bind all those subroutines with a real element from the optics file they must be registered by a call of `MARS_EL_REGISTER`, which takes the names of all those subroutines as arguments and save them into the internal data structures of MMBLB. For instance,

```
call mars_el_register(  
& 5,           ! ordinal number which identifies the element  
& nof_zones,  ! number of element non-standard zones  
& name_func,  ! subroutine that assigns type and name  
& init_func,  ! initialization  
& geo_func,   ! subroutine that describes geometry  
& mat_func,   ! subroutine that describes materials  
& field_func, ! subroutine that describes field  
& vol_func,   ! subroutine that calculates the zone volumes  
& zonename_func ) ! subroutine that associates name for each zone
```

In this example the names of subroutines are arbitrary. Actual names are chosen by the user and fixed at the registration with `MARS_EL_REGISTER`. A natural place for the element registration is in the subroutine `BLGEOINIT`, which is called at the initialization stage.

Let us describe the above subroutines in details (see also Section 8.3). The subroutine `name_func` tells MMBLB how to identify the element among all the others. The subroutine takes the type and name of the element as arguments and assigns character constants to them. As mentioned above, there are two ways to identify an element. The first way is to find the exact match for the type and name of the element in the OPTICS file. If this fails, then only the type is used for the search. If there are many elements in the beam line with the same geometry then the second way is preferable. In this case an element name should be used that for sure will not be found in the OPTICS file. A blank space can be used.

The subroutine `init_func` can be used for some calculations needed for an element initialization. For example, one can read in a field map from a file and save it into a program buffer. The subroutine is left dummy if no such calculations are needed.

The `geo_func` subroutine is meant for geometry description. The geometry is defined in a local frame of the element with the y -axis pointed upwards and the z -axis along the beam. The z -axis is a straight line connecting two points where a reference orbit enters and exits the element. The reference orbit enters the element at $(0, 0, 0)$ in the local coordinates. The subroutine returns a local zone number that corresponds to a point with the coordinates given in the local frame. The local zones are numbered from the unity. MMBLB translates the local zone number to a MARS non-standard zone number. The subroutine also returns a flag that is normally used for a beam enclosure description (see Section 7).

MMBLB imposes some restrictions on where elements can be defined. The front and rear surfaces of the elements in MAD can have an arbitrary orientation with respect to the

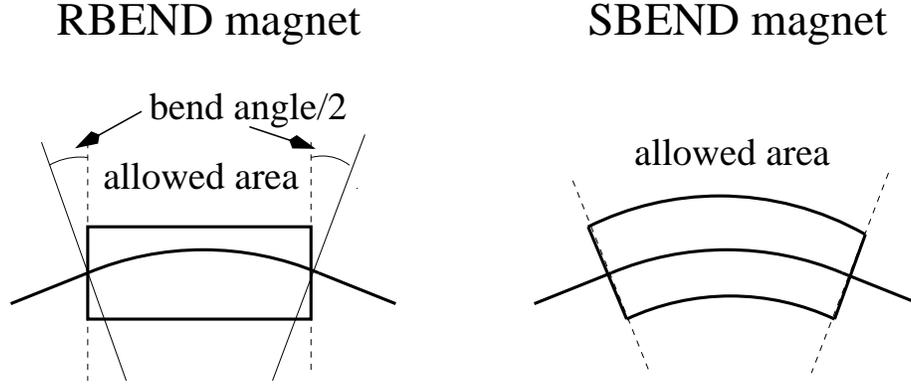


Figure 3: Areas of definition for RBEND and SBEND dipoles in MMBLB. The areas where the dipoles can be defined in MMBLB are restricted with the dashed lines.

reference orbit. The “area of definition” in MMBLB is usually limited to the two planes normal to the reference orbit at points where it enters and exits the element. The only exception is dipoles of the type “RBEND”. The area allowed for a rectangular dipole is formed with two parallel planes, as shown in Fig. 3.

The subroutine `field_func` takes the coordinates of a point in the local frame of the element and calculates the field in this point. The subroutines `mat_func`, `vol_func` and `zonename_func` return the material index, volume and zone name for a zone with the local index taken as an argument.

7 Beam Enclosure Description

In many cases, design of beam lines also includes design of radiation shielding. Shielding requirements are often driven by beam-induced effects such as residual dose on magnets and tunnel walls, ground water activation and prompt dose in occupied areas. Therefore, a complete model should include the tunnel walls followed by shielding.

The subroutine `TUNNEL_GEO` describes a beam enclosure in the local frame of a given element. The subroutine is especially appropriate in case where the beam enclosure follows the reference orbit so that its geometry can be defined in a simple unified way. The subroutine can only be used for planar 2-D beam lines, i.e. beam lines with no dipoles with non-zero roll angle. Skewed straight elements are acceptable since they do not change the reference orbit (Fig. 4). An interpretation of a tunnel geometry in the case of dipoles with non-zero roll angles is not obvious, so that such a situation is not handled in the current version. An attempt to use the subroutine in this case would lead to a misrepresentation of the tunnel geometry as shown in Fig. 5.

The subroutine determines a non-standard zone number for a point in the local frame of an element. It is automatically called by MMBLB every time when the point is found to be

in a slice of the beam enclosure associated with the given element, but not in the element itself. The given condition is emerged by a subroutine that processes the geometry of the element. The subroutine TUNNEL_GEO is called when the point (x,y,z) is not in the element (nz=0) but in a proper tunnel slice (lf=1). In a simple case the code can look as follows:

```

    subroutine elem_geo_func(length,
&      x,
&      Y,
&      z,
&      nz,
&      lf,
&      angle,
&      prevAngle,
&      nextAngle)
    implicit none

C...  INPUT parameters:
C      length      - length of element
C      x, y, z    - coordinates in the local frame
C      angle       - deflection angle of the element
C      prevAngle  - deflection angle of the previous element
C      nextAngle  - deflection angle of the next element

C...  OUTPUT parameters:
C      nz          - local zone number, will be transformed to
C                  non-standard zone number by the BLB
C      lf          - shows that the point (x,y,z) is in the
C                  tunnel slice associated with the element.
C                  1=yes, 0=no.

    integer nz, lf
    double precision length, x, y, z
    double precision angle, prevAngle, nextAngle

    nz = 0
    lf = 0

    if (z .ge. 0.0d0 .and. z .lt. length) then
        lf = 1
    end if

c...  the following code determines the zone number
    .....
```

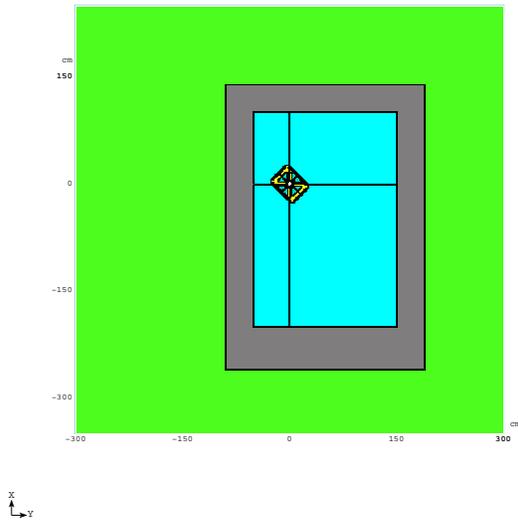


Figure 4: Beam enclosure around a skewed quadrupole, i.e. roll angle $\pi/4$.

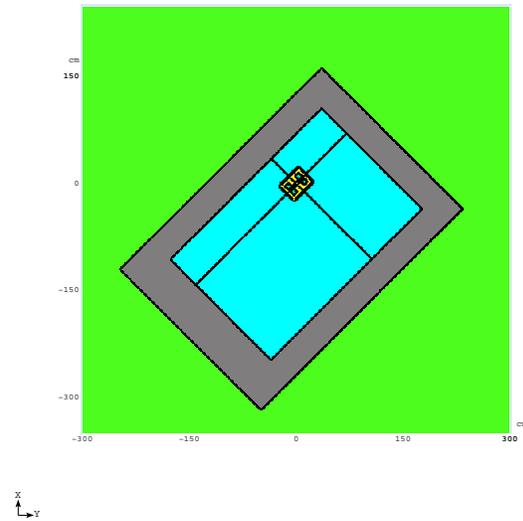


Figure 5: Beam enclosure around a dipole with non-zero roll angle.

A simple description of the tunnel slice, as shown in the example above, can lead to geometry misrepresentations. There will be unrealistic wedges in the tunnel in a case of “RBEND” magnets (Fig. 6). In order to correct for this misrepresentation the logical function INTUNNELSLICE is provided (Fig. 7). The function defines the tunnel slice taking into account the deflection angle of the element as well as the ones for the neighbours (Fig. 8). The function can be used for any element in the beam line but for sector magnets. The code in the example above should be changed as shown below for the function to be used.

```

C... old code
C   if (z .ge. 0.0d0 .and. z .lt. length) then
C     lf = 1
C   end if

   if( inTunnelSlice(length,
&     x,
&     y,
&     z,
&     angle,
&     prevAngle,
&     nextAngle)) then
     lf = 1
   end if

```

The explanation for the function arguments can be found in Section 8.

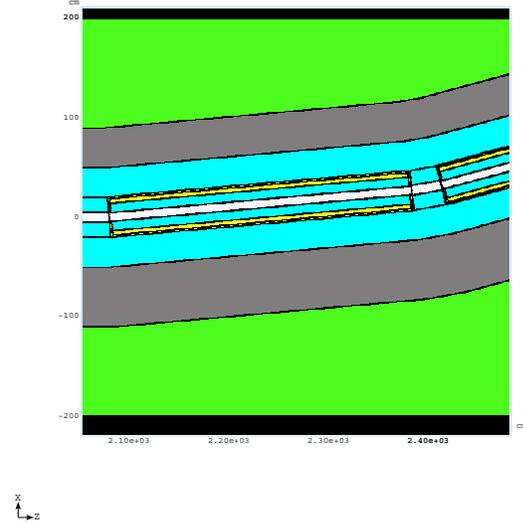
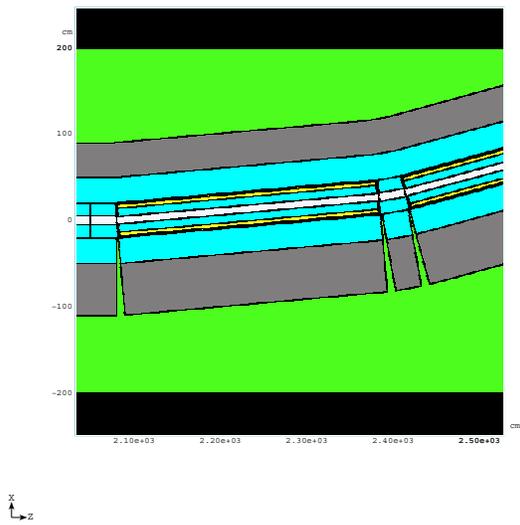


Figure 6: A simple description of the beam enclosure that leads to a geometry misrepresentation.

Figure 7: The use of INTUNNEL-SLICE corrects the geometry misrepresentation shown in Fig. 6.

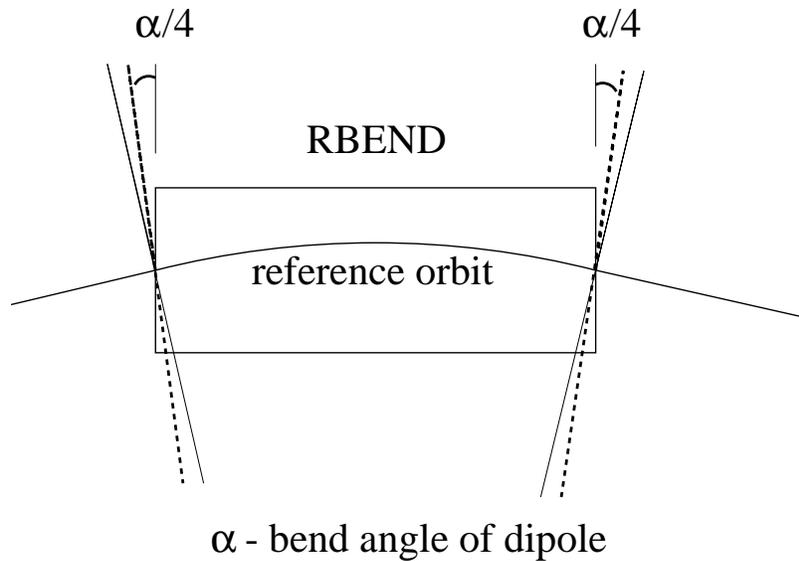


Figure 8: A tunnel slice for a RBEND type magnet as defined by the function INTUNNEL-SLICE. The tunnel slice is an area limited by the dashed lines.

From a practical standpoint, it is often useful to have longitudinal zone subdivision of beam enclosures. The subroutine TUNNEL_GEO provides a number of ways in which that can be done. In a real model all these ways can be used in one code. There are three examples given below.

Example 1: Uniform division along the s-axis.

This type of division is done with the TUNNEL_GEO arguments ZLOCMAD and ELBEGINS (see Section 8). An offset from the beginning of a beam line is determined as ZLOCMAD + ELBEGINS. In the example below, a sample of the TUNNEL_GEO subroutine is provided that demonstrates a division of concrete tunnel wall and a layer of soil into zones with a uniform length of 150 cm. The result of division is shown in Fig. 9.

```

SUBROUTINE TUNNEL_GEO(XLOCMAD, YLOCMAD, ZLOCMAD,
&    XGLMAD, YGLMAD, ZGLMAD, NZ, BLINDX, ELINDX,
&    ELBEGINS, ELLENGTH, BENDANGLE, BENDPREV, BENDNEXT)
  IMPLICIT NONE
  DOUBLE PRECISION XLOCMAD, YLOCMAD, ZLOCMAD,
&    XGLMAD, YGLMAD, ZGLMAD,
&    ELBEGINS, ELLENGTH,
&    BENDANGLE, BENDPREV, BENDNEXT
  INTEGER NZ, BLINDX, ELINDX
  DOUBLE PRECISION R, S
  INTEGER SLICE, ADD, ZONE
  R = SQRT(XLOCMAD**2 + YLOCMAD**2)
  S = ZLOCMAD + ELBEGINS
  SLICE = S / 150.0D0
  NZ = 0
  IF (R .LT. 100.0D0) THEN          ! AIR
    ADD = 1
    NZ = 1001
    RETURN
  ELSE IF (R .LT. 150.0D0) THEN ! CONCRETE WALL
    ADD = 2
  ELSE IF (R .LT. 200.0D0) THEN ! LAYER OF SOIL
    ADD = 3
  ELSE                               ! SOIL
    NZ = 1004
    RETURN
  END IF
  ZONE = 1000 + SLICE*10 + ADD
  NZ = ZONE
  RETURN
END

```

Example 2: Division according to the element index in a beam line.

This type of division is based on the index ELINDX of an element in a given beam line. The index is counted from the unity. In the example below, only one zone is assigned to the wall and the layer of soil for each tunnel slice. The result of this division is shown in Fig. 10.

```

SUBROUTINE TUNNEL_GEO(XLOCMAD, YLOCMAD, ZLOCMAD,
&    XGLMAD, YGLMAD, ZGLMAD, NZ, BLINDX, ELINDX,
&    ELBEGINS, ELLENGTH, BENDANGLE, BENDPREV, BENDNEXT)
IMPLICIT NONE
DOUBLE PRECISION XLOCMAD, YLOCMAD, ZLOCMAD,
&    XGLMAD, YGLMAD, ZGLMAD, ELBEGINS, ELLENGTH,
&    BENDANGLE, BENDPREV, BENDNEXT
INTEGER NZ, BLINDX, ELINDX
DOUBLE PRECISION R, S
INTEGER ADD, ZONE
R = SQRT(XLOCMAD**2 + YLOCMAD**2)
NZ = 0
IF (R .LT. 100.0D0) THEN      ! AIR
    ADD = 1
    NZ = 1001
    RETURN
ELSE IF (R .LT. 150.0D0) THEN ! CONCRETE WALL
    ADD = 2
ELSE IF (R .LT. 200.0D0) THEN ! LAYER OF SOIL
    ADD = 3
ELSE                          ! SOIL
    NZ = 1004
    RETURN
END IF
ZONE = 1000 + (ELINDX-1)*10 + ADD
NZ = ZONE
RETURN
END
```

Example 3: Division using global coordinates.

Some parts of a beam enclosure can have a complex geometry that can not be described in a simple unified way. In this case the arguments XGLMAD, YGLMAD and ZGLMAD of the subroutine TUNNEL_GEO prove to be useful. These arguments are coordinates in the global MAD frame. Such a geometry could, of cause, be described somewhere else in the MARS subroutines, but it makes sense to keep the tunnel description in one subroutine. Fig. 11 shows a model of the Tevatron $C\bar{O}$ interaction region of the BTeV experiment at Fermilab. The rightmost part of the enclosure is described with the use of the global coordinates.

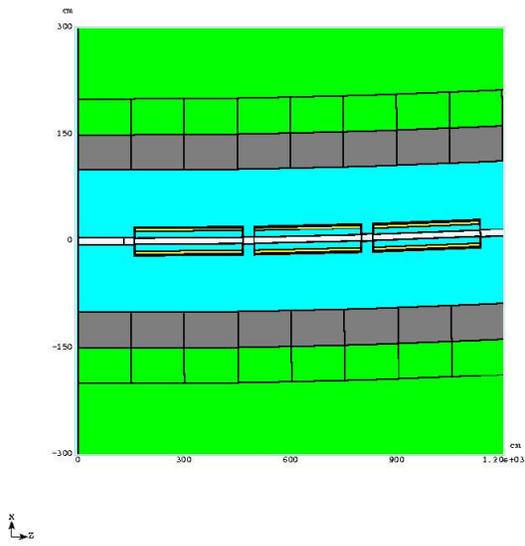


Figure 9: Beam enclosure division into zones with a uniform length of 150 cm.

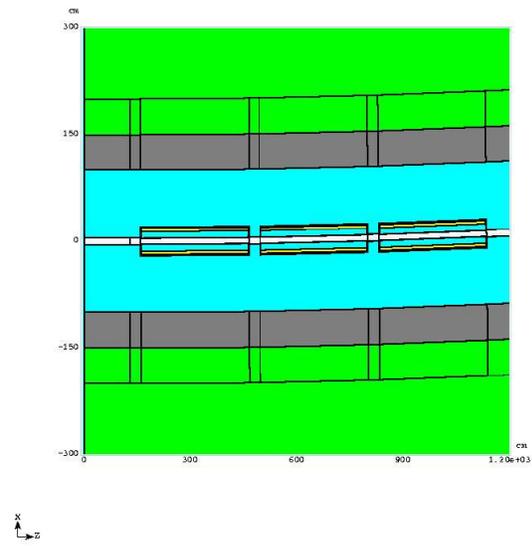


Figure 10: Beam enclosure division with only one zone corresponding to a tunnel slice.

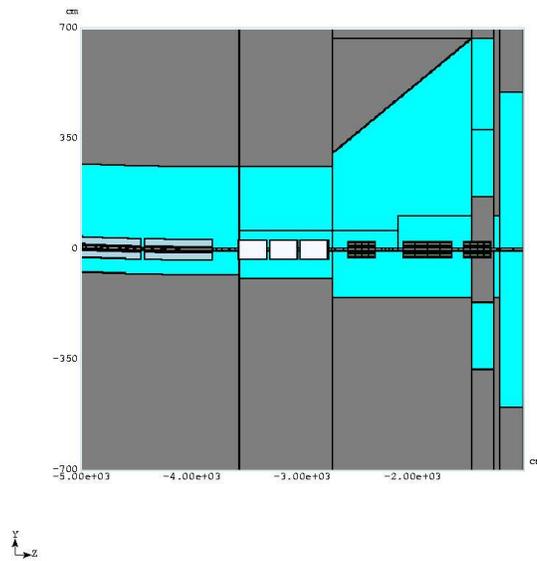


Figure 11: Tevatron $C\emptyset$ interaction region.

8 Summary of MMBLB Functions

The functions that come with the MMBLB libraries are described in this section. Parameters that change their values in the functions described below are marked with a star imprint. In the case when a parameter of the function is an array of a predefined size, the size is given in the parentheses after the parameter type.

8.1 “Hook” Functions

The standard MARS distribution package provides a set of “hook” functions. The source code of the functions is available to users and can be modified to describe beam lines and tunnel parts.

8.1.1 “Hook” Functions Called at Initialization

SUBROUTINE BLGEOINIT (BEAMID)

Action:

Register beam elements.

Input parameter:

BEAMID - INTEGER, beam line index.

This is the part where the Beam Line Builder becomes aware of the functions that are responsible for element description.

SUBROUTINE BLINIT (MMAX, IMUN*)

Action:

Create and read in beam lines, assigning proper materials and names to zones of the beam lines and associated tunnel.

Input parameter:

MMAX - INTEGER, number of non-standard zones.

Output parameter:

IMUN - INTEGER(MMAX+1), array of material indexes for non-standard zones.

8.1.2 “Hook” Functions Called at Calculation

SUBROUTINE BML2MARS (BMLVEC, BMLDIR, MARSVEC*, MARSDIR*)

Action:

Convert coordinates and direction of a particle track from a beam line frame to the MARS frame.

Input parameters:

- BMLVEC - DOUBLE PRECISION(3), coordinates in the beam line frame.
- BMLDIR - DOUBLE PRECISION(3), direction vector in the beam line frame.

Output parameters:

- MARSVEC - DOUBLE PRECISION(3), coordinates in the MARS frame.
- MARSDIR - DOUBLE PRECISION(3), direction vector in the MARS frame.

The function can be used instead of FLAT2GLOB whenever more sophisticated coordinate transformations are required. It may call the functions FLAT2GLOB and MAD2MARS.

SUBROUTINE MARS2BML (MARSVEC, MARSDIR ,BMLVEC*, BMLDIR*)

Action:

Convert coordinates and direction of a particle track from the MARS frame to a beam line frame.

Input parameters:

- MARSVEC - DOUBLE PRECISION(3), coordinates in the MARS frame.
- MARSDIR - DOUBLE PRECISION(3), direction vector in the MARS frame.

Output parameters:

- BMLVEC - DOUBLE PRECISION(3), coordinates in the beam line frame.
- BMLDIR - DOUBLE PRECISION(3), direction vector in the beam line frame.

The function can be used instead of GLOB2FLAT whenever more sophisticated coordinate transformations are required. It may call the functions GLOB2FLAT and MARS2MAD.

SUBROUTINE TUNNEL_GEO (XLOCMAD, YLOCMAD, ZLOCMAD,

XGLMAD, YGLMAD, ZGLMAD,
NZ*, BLINDEX, ELINDEX,
ELBEGINS, ELLENGTH,
BENDANGLE, BENDPREV, BENDNEXT)

Action:

Define the geometry of a tunnel associated with a beam line.

Input parameters:

XLOCMAD, YLOCMAD, ZLOCMAD	- DOUBLE PRECISION, coordinates in the local frame of an element. Normally, the element starts at ZLOCMAD=0.
XGLMAD, YGLMAD, ZGLMAD	- DOUBLE PRECISION, same coordinates in the global MAD frame.
BLINDEX	- INTEGER, index of a beam line, counted from the unity.
ELINDEX	- INTEGER, index of an element in the given beam line, counted from the unity.
ELBEGINS	- DOUBLE PRECISION, <i>s</i> -position of the beginning of the element.
ELLENGTH	- DOUBLE PRECISION, length of the element.
BENDANGLE	- DOUBLE PRECISION, bending angle of the element.
BENDPREV	- DOUBLE PRECISION, bending angle of the preceding element.
BENDNEXT	- DOUBLE PRECISION, bending angle of the following element.

Output parameter:

NZ	- INTEGER, zone number.
----	-------------------------

The subroutine is called from BLGEO. If no tunnel description is needed than the subroutine is left dummy.

8.2 Other MMBLB Functions

Unlike the “hook” functions, the ones described in this subsection come with the MMBLB library with the source code not available to the user.

8.2.1 Other MMBLB Functions Called at Initialization

SUBROUTINE BEAMLINES_IN_USE (NLINES)

Action:

Reserve memory for NLINES beam lines.

Input parameter:

NLINES - INTEGER, number of beam lines.

The subroutine must be called during the initialization before beam lines are read in from OPTICS files.

SUBROUTINE BLBUILD (OFNAME, X0, Y0, Z0, THETA, PHI, PSI, IZONE)

Action:

Build a beam line out of an OPTICS file.

Input parameters:

OFNAME - CHARACTER*(*), OPTICS file name.
X0, Y0, Z0 - DOUBLE PRECISION, coordinates of the beginning of the beam line.
THETA - DOUBLE PRECISION, angle between the z-axis and a projection of the beam line on horizontal plane.
PHI - DOUBLE PRECISION, elevation angle, i.e. an angle between the beam line direction and its projection on the horizontal plane.
PSI - DOUBLE PRECISION, roll angle around the beam line direction.
IZONE - first non-standard zone to be assigned to the beam line.

The parameters X0, Y0, Z0, THETA, PHI, PSI represent the initial position and orientation of the beam line.

SUBROUTINE BLMAT(IMUN*)

Action:

Fill appropriate elements of the integer array INUM with the material indices of the MARS non-standard zones assigned to the current beam line.

Output parameter:

IMUN - INTEGER, array of the material indexes of the non-standard zones.

The first element of INUM must correspond to the first non-standard zone.

SUBROUTINE BLNAME (NAMES*)

Action:

Fill appropriate elements of the array NAMES with the names of the non-standard zones assigned to the current beam line.

Output parameter:

NAMES - CHARACTER*8, array of zone names of the non-standard zones.

The first element of NAMES must correspond to the first non-standard zone.

SUBROUTINE BLSETE (BKINE, BMASS, CIRCUM)

Action:

Set parameters for a reference orbit in the current beam line.

Input parameters:

BKINE - DOUBLE PRECISION, kinetic energy of a particle on a reference orbit.
BMASS - DOUBLE PRECISION, mass of the particle.
CIRCUM - DOUBLE PRECISION, approximate length of the accelerator.

The parameter CIRCUM is purely informative. It is not used in any calculations. The parameters BKINE and BMASS are used to normalize the B-field in the beam line elements.

SUBROUTINE BLVOL (VOLUMES*)

Action:

Fill appropriate elements of the array VOLUMES with the volumes of the non-standard zones assigned to the current beam line.

Output parameter:

VOLUMES - DOUBLE PRECISION, array of volumes of the non-standard zones.

The first element of VOLUMES must correspond to the first non-standard zone.

SUBROUTINE MARS_EL_REGISTER(ELID, NZONES, NAMEFUNC, INITFUNC, GEOFUNC, MATFUNC, FIELDFUNC, VOLFUNC, ZONENAMEFUNC)

Action:

Register an element in a beam line.

Input parameters:

ELID	- INTEGER, unique element ID.
NZONES	- INTEGER, number of zones to be reserved for the given element.
NAMEFUNC	- name of subroutine that defines the type and name of the element.
INITFUNC	- name of initialization subroutine.
GEOFUNC	- name of subroutine that describes the geometry of the element.
MATFUNC	- name of subroutine that describes the materials used in the element.
FIELDFUNC	- name of subroutine that describes the B-field.
VOLFUNC	- name of subroutine that describes the volumes of zones.
ZONENAMEFUNC	- name of subroutine that describes the names of zones.

8.2.2 Other MMBLB Functions Called at Calculation Stage

SUBROUTINE BLFIELD(X, Y, Z, BX*, BY*, BZ*, ABSB*)

Action:

If the point (X,Y,Z) is in a non-standard zone of the current beam line, calculate the B-field in this point.

Input parameters:

X,Y,Z - DOUBLE PRECISION, coordinates in the mother frame.

Output parameters:

BX, BY, BZ - DOUBLE PRECISION, B-field in the mother frame.
 ABSB - DOUBLE PRECISION, magnitude of the B-field.

SUBROUTINE BLGEO(X, Y, Z, NZ*)

Action:

Find a non-standard zone number corresponding to the point (X,Y,Z). The search is only performed for the current beam line.

Input parameters:

X,Y,Z - DOUBLE PRECISION, coordinates in the mother frame.

Output parameter:

NZ - INTEGER, non-standard zone number or 0 if the point does not pertain to the current beam line.

SUBROUTINE BLPRINT (FILENAME)

Action:

Print information about each element in the current beam line into a file with the name FILENAME. The file is self-explanatory. It contains the element type, name, *s*-position, length, first and last MARS non-standard zone assigned to the element.

Input parameter:

FILENAME - CHARACTER*(*), name of the file.

SUBROUTINE BLZONES (ZONES*)

Action:

Retrieve the number of non-standard zones reserved for the current beam line.

Output parameter:

ZONES - INTEGER, number of zones.

SUBROUTINE FLAT2GLOB (POS*, DIR*)

Action:

Convert the position POS and direction DIR of a particle from a frame of beam line (X,Y,S) to the MARS global frame(X,Y,Z).

Output parameters:

POS - DOUBLE PRECISION(3), position in the beam line frame.

DIR - DOUBLE PRECISION(3), direction in the beam line frame.

If the *s*-coordinate exceeds the length of the beam line, the conversion is performed with the transformation matrix of the last element. Similarly, if *s* is smaller than the *s*-coordinate where the beam line begins then the transformation matrix of the first element is used.

SUBROUTINE GLOB2FLAT (POS*, DIR*)

Action:

Convert the position POS and direction DIR of a particle from the MARS global frame (X,Y,Z) to a frame of the beam line (X,Y,S).

Output parameters:

POS - DOUBLE PRECISION(3), position in the global frame.
DIR - DOUBLE PRECISION(3), direction in the global frame.

LOGICAL FUNCTION INTUNNELSLICE(LENGTH, LOCX, LOCY, LOCZ,
BENDANGLE, PREVBENDANGLE, NEXTBENDANGLE)

Action:

Determine whether or not the point (LOCX, LOCY, LOCZ) in a tunnel slice associated with a given element. The function returns .TRUE. if the answer is positive.

Input parameters:

LENGTH - DOUBLE PRECISION, length of the element.
LOCX, LOCY, LOCZ - DOUBLE PRECISION, coordinates in the local frame of the element.
BENDANGLE - DOUBLE PRECISION, bend angle of the element.
PREVBENDANGLE - DOUBLE PRECISION, bend angle of the previous element.
NEXTBENDANGLE - DOUBLE PRECISION, bend angle of the next element.

The function must not be used for sector magnets.

SUBROUTINE MAD2MARS (XMAD,YMAD,ZMAD,XMARS*,YMARS*,ZMARS*)

Action:

Convert coordinates from the traditional MAD frame to the MARS frame.

Input parameters:

XMAD, YMAD, ZMAD - DOUBLE PRECISION, coordinates in the MAD frame to be converted.

Output parameters:

XMARS, YMARS, ZMARS - DOUBLE PRECISION, resulting coordinates in the MARS frame.

SUBROUTINE MARS2MAD (XMARS,YMARS,ZMARS,XMAD*,YMAD*,ZMAD*)

Action:

Convert coordinates from the traditional MARS frame to the MAD frame.

Input parameters:

XMARS, YMARS, ZMARS - DOUBLE PRECISION, coordinates in the MARS frame to be converted.

Output parameters:

XMAD, YMAD, ZMAD - DOUBLE PRECISION, resulting coordinates in the MAD frame.

SUBROUTINE SET_CURRENT_BEAMLINE (LINEID)**Action:**

Set index of a beam line to work with.

Input parameter:

LINEID - INTEGER, index of the beam line. Counted from the unity.

8.3 Functions for Element Definition

The functions described below may have arbitrary names. The actual names are given at the time of element registration by invoking the function MARS_EL_REGISTER. The functions appear in the same order as in the argument list of MARS_EL_REGISTER.

SUBROUTINE NAMEFUNC(TYPE*, NAME*)**Action:**

Define the type and name of the element. The type and name are used to search for the element in the MMBLB internal data structures.

Output parameters:

TYPE - CHARACTER*80, type of the element.

NAME - CHARACTER*80, name of the element.

SUBROUTINE INITFUNC(TYPE, NAME, LENGTH, POS, FIELDS)**Action:**

The subroutine is a natural place to perform the initialization. For example, fill data structures for the geometry, fields and so on.

Input parameters:

TYPE	- CHARACTER*80, type of the element.
NAME	- CHARACTER*80, name of the element.
LENGTH	- DOUBLE PRECISION, length of the element.
POS	- DOUBLE PRECISION, <i>s</i> -position of the beginning of the element.
FIELDS	- DOUBLE PRECISION(4), dipole, quadrupole, sextupole and octupole field components.

SUBROUTINE GEOFUNC(LENGTH, X, Y, Z, NZ*, LF*,
ANGLE, PREVANGLE, NEXTANGLE)

Action:

Determine a local zone number NZ corresponding to the point (X, Y, Z). NZ must be set to zero if the point (X, Y, Z) does not pertain to the element. The flag LF is set to the unity if the point (X, Y, Z) is in a tunnel slice of the given element. LF is set to zero otherwise.

Input parameters:

LENGTH	- DOUBLE PRECISION, length of the element.
X, Y, Z	- DOUBLE PRECISION, coordinates in the local frame of the element.
ANGLE	- DOUBLE PRECISION, deflection angle of the element.
PREVANGLE	- DOUBLE PRECISION, deflection angle of the previous element, 0 if the given element is the first one in a beam line.
NEXTANGLE	- DOUBLE PRECISION, deflection angle of the next element, 0 if the given element is the last one in a beam line.

Output parameters:

NZ	- INTEGER, local zone number (counted from the unity).
LF	- INTEGER, flag that shows whether or not the point (X, Y, Z) is in a tunnel slice of the given element.

MMBLB launches the subroutine TUNNEL_GEO in the case of a special situation when NZ is zero and LF is the unity.

SUBROUTINE MATFUNC (NZ, IM*)

Action:

Return an index of the material IM corresponding to the local zone NZ.

Input parameter:

NZ	- INTEGER, local zone number (counted from the unity).
----	--

Output parameter:

IM - INTEGER, index of the material that corresponds to the local zone NZ.

SUBROUTINE FIELDFUNC(LENGTH, FIELDS, X, Y, Z, BX*, BY*, BZ*, ABSB*,
ANGLE)

Action:

Calculate the B-field in the point (X, Y, Z).

Input parameters:

LENGTH - DOUBLE PRECISION, length of the element.
FIELDS - DOUBLE PRECISION(4), dipole, quadrupole, sextupole and octupole field components.
X, Y, Z - DOUBLE PRECISION, coordinates in the local frame of the element.
ANGLE - DOUBLE PRECISION, deflection angle of the element.

Output parameters:

BX, BY, BZ - DOUBLE PRECISION, (X,Y,Z)-components of the B-field.
ABSB - DOUBLE PRECISION, absolute magnitude of the B-field.

SUBROUTINE VOLFUNC(NZ, LENGTH, VOLUME*, ANGLE)

Action:

Return a volume of the local zone NZ.

Input parameters:

NZ - INTEGER, local zone number (counted from the unity).
LENGTH - DOUBLE PRECISION, length of the element.
ANGLE - DOUBLE PRECISION, deflection angle of the element.

Output parameter:

VOLUME - DOUBLE PRECISION, volume of the local zone NZ.

SUBROUTINE ZONENAMEFUNC(NZ, ZNAME*)

Action:

Return a name corresponding to the local zone NZ.

Input parameter:

NZ - INTEGER, local zone number (counted from the unity).

Output parameter:

ZNAME

- CHARACTER*8, zone name corresponding to the local zone
NZ.

9 Acknowledgments

We are thankful to L.P. Michelotti, D.N. Mokhov and J.-F. Ostiguy for contributions to the earlier version of MMBLB , and to A.D. Russell for fruitful discussions and constructive comments.

References

- [1] N.V. Mokhov, “The MARS Code System User’s Guide”, Fermilab-FN-628 (1995); N.V. Mokhov, O.E. Krivosheev, “MARS Code Status”, in *Proc. Monte Carlo 2000 Conf.*, pp. 943-948, Lisbon, October 23-26, 2000; Fermilab-Conf-00/181 (2000); N.V. Mokhov, “Status of MARS Code”, Fermilab-Conf-03/053 (2003); <http://www-ap.fnal.gov/MARS/>.
- [2] N.V. Mokhov et al., “Recent Enhancements to the MARS15 Code”, FERMILAB-Conf-04/053-AD, April 2004; Presented paper at the 10th *International Conference on Radiation Shielding*, Funchal (Madeira), Portugal, May 9-14, 2004. e-Print Archive: nuclth/0404084.
- [3] H. Grote and F.C. Iselin, “The MAD program (Methodical Accelerator Design)”, CERN/SL/90-13 (1990).
- [4] D.N. Mokhov et al., “MAD Parsing and Conversion Code”, Fermilab-TM-2115 (2000).
- [5] O.E. Krivosheev et al., “A Lex-based MAD parser and its applications”, in *Proc. 2001 Particle Accelerator Conference*, pp. 3036-3038, Chicago, IL, June 18-22, 2001; Fermilab-Conf-01/142-T (2001).
- [6] A.I. Drozhdin et al., “Beam loss, residual radiation, and collimation and shielding in the Fermilab booster,” in *Proc. 2001 Particle Accelerator Conference*, pp. 2569-2571, Chicago, IL, June 18-22, 2001; Fermilab-Conf-01/141 (2001).
- [7] N.V. Mokhov et al., “Fermilab booster beam collimation and shielding,” in *Proc. 2003 Particle Accelerator Conference*, Portland, Oregon, May 12-16, 2003; Fermilab-Conf-03/087 (2003).
- [8] A.I. Drozhdin, O.E. Krivosheev and N.V. Mokhov, “Beam loss and collimation in the Fermilab 16-GeV proton driver,” in *Proc. 2001 Particle Accelerator Conference*, pp. 2572-2574, Chicago, IL, June 18-22, 2001; Fermilab-Conf-01/128 (2001).
- [9] A.I. Drozhdin, O.E. Krivosheev and N.V. Mokhov, “Beam loss, collimation and shielding at the Fermilab Proton Driver,” Fermilab-FN-693 (2000).
- [10] I.S. Tropin, N.V. Mokhov and S. Childress, “Consequences of accidental beam loss in the NuMI primary beam line”, Fermilab NuMI-Note-Beam-817 (2002).
- [11] M.A. Kostin et al., “Radiation in the NuMI stub at beam accidents in the Main Injector”, Fermilab NuMI-Note-Sim-889 (2002).
- [12] N. Nakao et al., “MARS14 Shielding Calculations for the J-PARC 3 GeV RCS”, KEK Report 2004-1, June 2004.
- [13] <http://laacg1.lanl.gov/laacg/services/possup.html>
- [14] <http://www.vectorfields.com>