AT 2.0 Manual

September 2, 2015

Abstract

We give an overview for the AT 2.0 code and describe how the functions and repository fit into this.

1 Introduction

AT is a toolbox of functions in Matlab for charged particle beam simulation. On the one hand, as a toolbox, one should be allowed to use the tools to do what one wants. On the other, in order to avoid redundancy in work, and facilitate working together via sharing of tools, it is useful to implement some standards. We describe the tools and some standards and guidelines in AT with the hope that the overall package has some coherence and elegance (speed is also considered at some points where it is critical).

2 Physics

2.1 Coordinates

The 6-d phase space coordinates are as follows

$$\vec{Z} = \begin{pmatrix} x \\ \frac{p_x}{P_0} \\ y \\ \frac{p_y}{P_0} \\ \delta \\ ct \end{pmatrix} \tag{1}$$

The momenta are defined as

$$x' = \frac{p_x}{P_z} \quad y' = \frac{p_y}{P_z} \tag{2}$$

with $P_z = P_0(1 + \delta)$.

3 Lattice Creation

A lattice may be created in a number of ways. The final result should be a cell array whose elements are the lattice elements.

There are number of different element "classes". These are Drift, Bend, Quadrupole, Sextupole, Octupole, Multipole, ThinMultipole, Wiggler, KickMap, RFCavity, QuantDiff, Monitor, Corrector, Solenoid, Matrix66, RingParam.

4 Pass Methods

Each element needs a pass method which takes the phase space coordinate of the electron before the element and returns the phase space coordinate after. In the absence of radiation, the result will be symplectic. The pass methods are so-called *symplectic integrators*. The integrators available in AT are BndMPoleSymplectic4E2Pass, BndMPoleSymplectic4E2RadPass, BndMPoleSymplectic4Pass, BndMPoleSymplectic4RadPass, BendLinearPass, CavityPass, QuadLinearPass, QuadMPoleFringePass, StrMPoleSymplectic4Pass, StrMPoleSymplectic4RadPass, StrMPoleSymplectic4RadPass, WiggLinearPass, IDTablePass,

A 4th order symplectic integrator assumes that the Hamiltonian can be written in the form $H = H_1 + H_2$ and that H_1 and H_2 can be independently solved.

The Hamiltonian may be written

$$H = 1 + \delta - (1 + hx)\frac{A_s}{B\rho} - (1 + hx)\sqrt{(1 + \delta)^2 - p_x^2 - p_y^2} \tag{3}$$

and this can be expanded and split. Now, given a splitting, we define

$$d_1 = d_4 = \frac{1}{2 - 2^{1/3}}L \tag{4}$$

$$d_2 = d_3 = \frac{1 - 2^{1/3}}{2(2 - 2^{1/3})}L \tag{5}$$

$$k_1 = k_3 = \frac{1}{2 - 2^{1/3}}L \tag{6}$$

$$k_2 = -\frac{2^{1/3}}{2 - 2^{1/3}}L \tag{7}$$

Now, suppose we can solve H_1 (drift) and H_2 (kick) independently, and we notate

$$e^{:H_1 d:} = D(d) \tag{8}$$

$$e^{:H_2 :k} = K(k) \tag{9}$$

Then the 4th order integrator is

$$D(d_1)K(k_1)D(d_2)K(k_2)D(d_2)K(k_1)D(d_1) \tag{10}$$

One possibility is (improve this)

$$H_1 = (1+hx)\frac{p_x^2+p_y^2}{2(1+\delta)} \tag{11}$$

$$H_2 = -(1+hx)\frac{A_s}{B\rho} - (1+\delta)hx \tag{12}$$

where $h = 1/\rho$ with ρ the bending radius of the magnet (for a given energy electron).

5 Working with Lattices

There are two kinds of functions for working with lattices. One type asks a question about certain kinds of elements and returns a set of indices to those elements. The function *atgetcells* is an important example of this type which asks for field names in the lattice structure and looks for matches. The output is a set of boolean values corresponding to whether or not each element matches the criterion. One can also use the Matlab function *findcells* in the same way. The output here is a list of element indices. (getcellstruct and setcellstruct?)

5.1 atgetfieldvalues

ATGETFIELDVALUES retrieves the field values AT cell array of elements

VALUES = ATGETFIELDVALUES(RING,'field') extracts the values of the field 'field' in all the elements of RING

VALUES = ATGETFIELDVALUES(RING,INDEX,'field') extracts the values of the field 'field' in the elements of RING selected by INDEX

if RINGI.FIELD is a numeric scalar VALUES is a length(INDEX) x 1 array otherwise VALUES is a length(INDEX) x 1 cell array

More generally ATGETFIELDVALUES(RING,INDEX,subs1,subs2,...) will call GETFIELD(RINGI,subs1,subs2,...) for I in INDEX

Examples:

V=ATGETFIELDVALUES(RING,1:10,'PolynomB') is a 10x1 cell array such that VI=RINGI.PolynomB for I=1:10

V=ATGETFIELDVALUES(RING(1:10),'PolynomB',1,2) is a 10x1 array such that V(I)=RINGI,PolynomB(1,2)

5.2 atsetfieldvalues

ATSETFIELDVALUES sets the field values of MATLAB cell array of structures

Note that the calling syntax must be in the form of assignment: RING = ATSETFIELDVALUES(RING,...) MATLAB does not modify variables that only appear on the right hand side as arguments.

NEWRING=ATSETFIELDVALUES(RING,'field',VALUES) In this mode, the function will set values on all the elements of RING

NEWRING=ATSETFIELDVALUES(RING,INDEX,'field',VALUES) In this mode, the function will set values on the elements of RING specified by INDEX, given as a list of indices or as a logical mask

NEWRING=ATSETFIELDVALUES(RING,'field',VALUESTRUCT) In this mode, the function will set values on the elements of RING whose family names are given by the field names of VALUESTRUCT

NEWRING=ATSETFIELDVALUES(RING,RINGINDEX,....,VALUESTRUCT) As in the previous mode, the function will set values on the elements of RING whose family names are given by the field names of VALUESTRUCT. But RINGINDEX=atindex(RING) is provided to avoid multiple computations.

Field selection ———————————————— NEWRING = ATSETFIELD(RING,'field',VALUES) For each I=1:length(RING), set RINGI.FIELD=value

NEWRING = ATSETFIELD(RING,'field',M,N,VALUES) For each I=1:length(RING), set RINGI.FIELD(M,N)=value

More generally, NEWRING = ATSETFIELD(RING,subs1,subs2,...,VALUES) For each I=1:length(RING), SETFIELD(RINGI,subs1,subs2,...,value)

The last dimension of VALUES must be either length(INDEX) or 1 (the value will be repeated for each element). For a vector to be repeated, enclose it in a cell array.

Value format ———————————————— Cell array VALUES ————— Mx1 cell array : one cell per element 1x1 cell array : cell 1 is affected to all selected elements

Character array VALUES ————————— 1xN char array (string) : the string as affected to all selected elements MxN char array : one row per element

Numeric array VALUES ————————— 1x1 (scalar) : the value is affected to all selected elements Mx1 (column) : one value per element MxN (matrix) : one row affected per element. If M==1, the single row is affected to all selected elements To assign column vectors to parameters, use a cell array VALUES, with one column per cell

A lattice manipulation function takes a lattice as an argument and produces a new lattice as a result. Here are some lattice manipulation functions:

5.3 Adding errors

- atsetshift
- atsettilt
- atsetfieldvalues
- atsplitdrift
- ataddmpolecomppoly
- ataddmpoleerrors
- atloadfielderrs

6 Programming guide

The tracking engine in AT is a Matlab interface to C code. The integrator (or pass method) is written in C, and is called by Matlab via the Mex interface.

Consider the following C code for tracking through an empty space (drift)

```
void DriftPass(double *r_in, double le, int num_particles)
/* le - physical length
   r_in - 6-by-N matrix of initial conditions reshaped into
   1-d array of 6*N elements
*/
{
    double *r6;
    int c;
double p_norm, NormL;

for (c = 0; c<num_particles; c++) { /*Loop over particles  */
        r6 = r_in+c*6;
        if(!mxIsNaN(r6[0])) {
            p_norm = 1/(1+r6[4]);
            NormL = le*p_norm;
            r6[0] += NormL*r6[1];
            r6[2] += NormL*r6[3];
            r6[5] += NormL*p_norm*(r6[1]*r6[1]+r6[3]*r6[3])/2;
        }
    }
}
```

This C function takes a set of input phase space vectors, a double indicating the length of the drift and an integer representing the number of particles. Note the use of the mxIsNaN function that comes from the mex.h header file.

The other requirements for a pass method are a MexFunction and a passFunction. The passFunction is defined as

```
ExportMode int* passFunction(const mxArray *ElemData,int *FieldNumbers,
                             double *r_in, int num_particles, int mode)
```

and the MexFunction as

```
void mexFunction( int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
```

The pass methods have two different calling methods. They may be called directly via the Mex interface (through the MexFunction entry point in the C function), or they may be called indirectly through the function RingPass (through the passFunction entry). The pass methods should be defined so that calling them with no arguments gives a list of required and optional parameters.

The moment tracking and equilibrium finding occurs via the function OhmiEnvelope(). For this to work requires pass methods that include radiation (this

gives a deterministic effect which results in damping and non-symplecticity. Further, the function findmpoleraddiffmatrix is required to compute the diffusion matrix.

7 Local and Global parameters

Given the ability to track particles through the lattice, one can compute global beam dynamics parameters and properties that vary around the ring. Here are some global parameters.

7.1 tunes

7.2 chromaticity

7.3 momentum compaction factor

7.4 tune shift with amplitude

Here are some quantities that vary around the ring and come from particle tracking of one turn:

7.5 Closed orbit

7.6 One turn map matrix

7.7 Transfer matrix from one position to another

7.8 Twiss Parameters

7.9 Dispersion function

From many turn tracking and determination of stability, one may compute at various points around the ring:

7.10 dynamic aperture: atdynap

7.11 momentum aperture: atmomentumaperture

From tracking the diffusion matrix and inclusion of radiation, one can compute:

7.12 Beam sizes

8 Lattice change and optimization from beam parameters

In addition to the lattice building tools in Section 5, there are also lattice changes one would like to do based on beam parameters. For example, one would like

to change the tunes, chromaticities or beta functions of the lattice.

- atfittune
- atfitchrome

For general optimization, one may use the atmatch routine.

8.1 atmatch

See documentation by S. Liuzzo.

9 Visualization

The lattice functions described in the previous section may be plotted, together with a synoptic representation of the lattice. The function atplot is designed for this purpose.

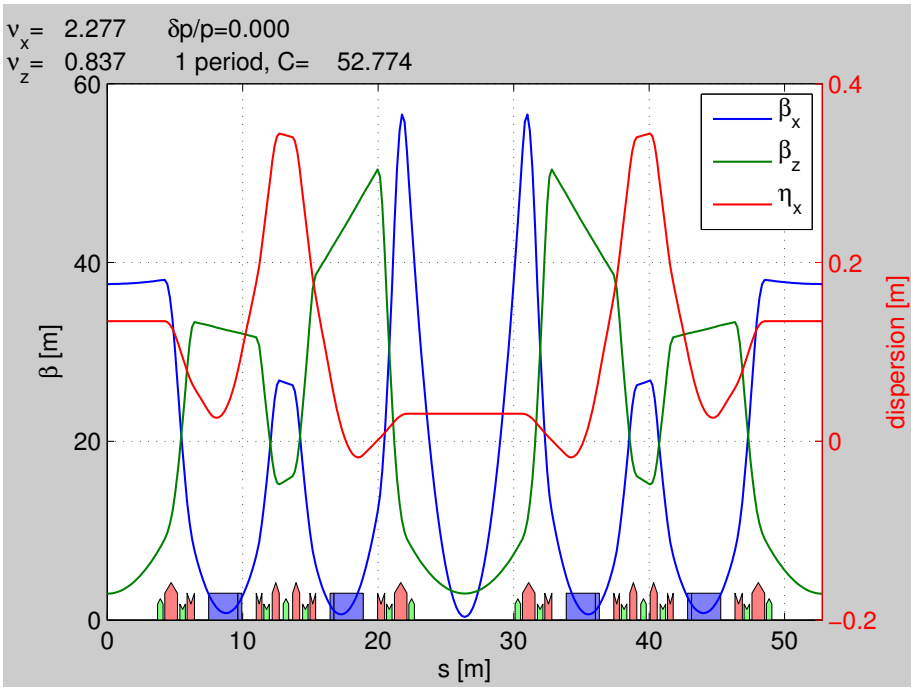

Figure 1: ESRF lattice from atplot

10 AT within a larger context: Other Codes, Matlab Middle Layer

11 Appendix: List of all functions and their help

atlinopt.m
ATLINOPT performs linear analysis of the COUPLED lattices

LinData = ATLINOPT(RING,DP,REFPTS) is a MATLAB structure array with fields

ElemIndex - ordinal position in the RING
SPos - longitudinal position [m]
ClosedOrbit - closed orbit column vector with
components x, px, y, py (momentums, NOT angles)
Dispersion - dispersion orbit position vector with
components eta_x, eta_prime_x, eta_y, eta_prime_y
calculated with respect to the closed orbit with
momentum deviation DP. Only if chromaticity is required.
M44 - 4x4 transfer matrix M from the beginning of RING
to the entrance of the element for specified DP [2]
A - 2x2 matrix A in [3]
B - 2x2 matrix B in [3]
C - 2x2 matrix C in [3]
gamma - gamma parameter of the transformation to eigenmodes
mu - [mux, muy] horizontal and vertical betatron phase
beta - [betax, betay] vector
alpha - [alphax, alphay] vector

All values are specified at the entrance of each element specified in REFPTS.
REFPTS is an array of increasing indexes that select elements
from the range 1 to length(LINE)+1.
See further explanation of REFPTS in the "help" for FINDSPOS

[LinData,NU] = LINOPT() returns a vector of linear tunes [nu_u , nu_v] for two normal modes of linear motion [1]

[LinData,NU, KSI] = LINOPT() returns a vector of chromaticities ksi = d(nu)/(dP/P)

$$ksiu, ksiv$$

- derivatives of

$$nuu, nuv$$

LinData = LINOPT(RING,DP,REFPTS,ORBITIN) does not search for closed orbit.
instead ORBITIN is used

Difference with linopt: Fractional tunes 0¡=tune¡1
Dispersion output (if chromaticity is required)
Alpha output
Phase advance output
Option to skip closed orbit search

See also ATREADBETA ATX ATMODUL FINDSPOS TWISSRING TUNECHROM

1

D.Edwars,L.Teng IEEE Trans.Nucl.Sci. NS-20, No.3, p.885-888, 1973

2

E.Courant, H.Snyder

3

D.Sagan, D.Rubin Phys.Rev.Spec.Top.-Accelerators and beams, vol.2 (1999)

atradon.m
ATRADON switches RF and radiation on

RING2=ATRADON(RING,CAVIPASS,BENDPASS,QUADPASS)

RING: initial AT structure
CAVIPASS: pass method for cavities (default ThinCavityPass)
BENDPASS: pass method for cavities (default BndMPoleSymplectic4RadPass)
QUADPASS: pass method for cavities (default: nochange)

$$RING2, RADINDEX, CAVINDEX$$

=ATRADON(...) returns the index of radiative elements
and cavities

atx.m
ATX computes and displays global information

BEAMDATA=ATX(RING,DPP,REFPTS)

RING: AT structure
DPP: relative energy deviation (default: 0)
REFPTS: Index of elements (default: 1:length(ring))

BEAMDATA is a MATLAB structure array with fields

From atlinopt:

ElemIndex - ordinal position in the RING
SPos - longitudinal position [m]
ClosedOrbit - closed orbit column vector with
components x, px, y, py (momentums, NOT angles)
Dispersion - dispersion orbit position vector with
components eta_x, eta_prime_x, eta_y, eta_prime_y
calculated with respect to the closed orbit with
momentum deviation DP
M44 - 4x4 transfer matrix M from the beginning of RING
to the entrance of the element for specified DP [2]
A - 2x2 matrix A in [3]
B - 2x2 matrix B in [3]
C - 2x2 matrix C in [3]
gamma - gamma parameter of the transformation to eigenmodes
mu - [mux, muy] horizontal and vertical betatron phase
beta - [betax, betay] vector
alpha - [alphax, alphay] vector

From ohmienvelope:

beam66 - 6x6 equilibrium beam matrix
emit66 - 6x6 emittance projections on x and y + energy spread
beam44 - intersection of beam66 for dpp=0
emit44 - emittances of the projections of beam44 on x and y
modemit - [emitA emitB] emittance of modes A and B (should be constant)

$$BEAMDATA, PARAMS$$

=ATX(...) Returns also a structure PM with fields
ll - Circumference
alpha - momentum compaction factor
nuh - Tunes
nuv
fulltunes
fractunes
espread - Energy spread

blength - Bunch length
modemittance - Eigen emittances

See also: ATREADBETA ATLINOPT OHMIENVELOPE ATMODUL

12 Appendix

12.1 Drift

- pass method: DriftPass
- creation function: ATDRIFT(FAMNAME,LENGTH,PASSMETHOD) creates a drift space element with Class 'Drift' FAMNAME family name LENGTH length [m] PASSMETHOD tracking function, defaults to 'DriftPass'

12.2 Bend

- pass methods: BndMPoleSymplectic4Pass, BndMPoleSymplectic4RadPass BndMPoleSymplectic4E2Pass, BndMPoleSymplectic4E2RadPass BndMPoleSymplectic4FrgFPass, BndMPoleSymplectic4FrgFRadPass BendLinearPass
- element creation functions ATRBEND(FAMNAME,LENGTH,BENDINGANGLE,K,PASSMETHOD) creates a rectangular bending magnet element with class 'Bend' FAMNAME family name LENGTH length of the arc for an on-energy particle [m] BENDINGANGLE total bending angle [rad] K focusing strength, defaults to 0 PASSMETHOD tracking function, defaults to 'BendLinearPass'

  ATRBEND(FAMNAME,LENGTH,BENDINGANGLE,K,PASSMETHOD,'FIELDNAME1',VALUE1,...) Each pair {'FIELDNAME',VALUE} is added to the element

  ATSBEND(FAMNAME,LENGTH,BENDINGANGLE,K,PASSMETHOD) creates a rectangular bending magnet element with class 'Bend' FAMNAME family name LENGTH length of the arc for an on-energy particle [m] BENDINGANGLE total bending angle [rad] K focusing strength, defaults to 0 PASSMETHOD tracking function, defaults to 'BendLinearPass'

  ATSBEND(FAMNAME,LENGTH,BENDINGANGLE,K,PASSMETHOD,'FIELDNAME1',VALUE1,...) Each pair {'FIELDNAME',VALUE} is added to the element

12.3 Quadrupole

- pass methods StrMPoleSymplectic4Pass, StrMPoleSymplectic4RadPass QuadLinearPass, QuadMPoleFringePass, QuadMPoleFringeRadPass ThinMPolePass

- element creation function ATQUADRUPOLE(FAMNAME,LENGTH,K,PASSMETHOD) creates a quadrupole element with Class 'Quadrupole'

  FAMNAME family name LENGTH length [m] K strength [m-2] PASSMETHOD tracking function, defaults to 'QuadLinearPass'

  ATQUADRUPOLE(FAMNAME,LENGTH,K,PASSMETHOD,'FIELDNAME1',VALUE1,...) Each pair 'FIELDNAME',VALUE is added to the element

12.4 Sextupole

- pass methods StrMPoleSymplectic4Pass, StrMPoleSymplectic4RadPass ThinMPolePass

- element creation function ATSEXTUPOLE(FAMNAME,LENGTH,S,PASSMETHOD) creates a sextupole element with class 'Sextupole'

  FAMNAME family name LENGTH length [m] S strength [m-2] PASSMETHOD tracking function, defaults to 'StrMPoleSymplectic4Pass'

12.5 Octupole

- element creation function atoctupole

- pass methods StrMPoleSymplectic4Pass, StrMPoleSymplectic4RadPass ThinMPolePass

12.6 Multipole

- pass methods StrMPoleSymplectic4Pass, StrMPoleSymplectic4RadPass ThinMPolePass

- element creation function ATMULTIPOLE(FAMNAME,LENGTH,POLYNOMA,POLYNOMB,PASSME creates a multipole element FAMNAME family name LENGTH length[m] POLYNOMA skew [dipole quad sext oct]; POLYNOMB normal [dipole quad sext oct]; PASSMETHOD tracking function. Defaults to 'StrMPoleSymplectic4Pass'

12.7 ThinMultipole

ATTHINMULTIPOLE(FAMNAME,POLYNOMA,POLYNOMB,PASSMETHOD) creates a thin multipole element with Class 'ThinMultipole'

FAMNAME family name POLYNOMA skew [dipole quad sext oct]; POLYNOMB normal [dipole quad sext oct]; PASSMETHOD tracking function. Defaults to 'ThinMPolePass'

12.8 Wiggler

- pass methods GWigSymplecticPass
- element creation function atwiggler(fname, Ltot, Lw, Bmax, Nstep, Nmeth, By, Bx, method)

  FamName family name Ltot total length of the wiggle Lw total length of the wiggle Bmax peak wiggler field [Tesla] Nstep num of integration steps per period Nmeth symplectic integration method, 2nd or 4th order: 2 or 4 By wiggler harmonics for horizontal wigglers Bx wiggler harmonics for vertical wigglers method name of the function to use for tracking

  returns a wiggler structure with class 'Wiggler'

12.9 KickMap

- pass methods IDTablePass
- element creation function atidtable(FAMNAME,Nslice,filename,Energy,method)

  FamName family name Nslice number of slices (1 means the wiggler is represented by a single kick in the center of the device). filename name of file with wiggler tracking tables. Energy Energy of the machine, needed for scaling method tracking function. Defaults to 'IdTablePass'

  The tracking table is described in P. Elleaume, "A new approach to the electron beam dynamics in undulators and wigglers", EPAC92.

  returns assigned structure with class 'KickMap'

12.10 RFCavity

- pass methods CavityPass
- element creation function ATRFCAVITY(FAMNAME,LENGTH,VOLTAGE,FREQUENCY,HARMONI
  creates an rfcavity element with Class 'RFCavity'

  FamName family name Length length[m] Voltage peak voltage (V) Frequency RF frequency [Hz] HarmNumber Harmonic Number Energy Energy in eV PassMethod name of the function on disk to use for tracking

  Also, use atsetcavity to set the cavity voltage with and without radiation.

```
%atsetcavity(ring,rfv, radflag,HarmNumber)
%sets the synchronous phase of the cavity assuming radiation is turned on
%radflag says whether or not we want radiation on, which affects
%synchronous phase.
%also sets the rf voltage and Harmonic number
%also sets the rf frequency.
```

The synchronous phase with radiation should be

$$\phi_s = sin^{-1}\left(\frac{U_0}{eV_{RF}}\right) \tag{13}$$

12.11 QuantDiff

- pass methods QuantDiffPass
- element creation function atQuantDiff(fname,ring)

12.12 Monitor

- pass methods IdentityPass
- element creation function ATMONITOR(FAMNAME)

12.13 Corrector

- pass methods CorrectorPass
- element creation function ATCORRECTOR(FAMNAME,LENGTH,KICK,PASSMETHOD)

12.14 Solenoid

- pass methods SolenoidLinearPass
- element creation function z=atsolenoid('FAMILYNAME',Length [m],KS,'METHOD') creates a new solenoid element with Class 'Solenoid' The structure with field FamName family name Length length[m] KS solenoid strength KS [rad/m] PassMethod name of the function to use for tracking

12.15 Matrix66

- pass methods Matrix66Pass
- element creation function atM66(fname,m66)

12.16 RingParam

ATRINGPARAM(rname,E0,per) creates a RingParameter Element which should go at the beginning of the ring
FNAME Family name which may be used as name of Ring Energy Energy of electrons PER Periodicity of the ring (=1 if ring is already expanded)

12.17 Integrator arguments

12.18 BndMPoleSymplectic4E2Pass

required arguments:
'Length' 'BendingAngle' 'EntranceAngle' 'ExitAngle' 'PolynomB' 'MaxOrder' 'NumIntSteps'
optional arguments:
'FullGap' 'FringeInt1' 'FringeInt2' 'H1' 'H2' 'T1' 'T2' 'R1' 'R2'

12.19 BndMPoleSymplectic4E2RadPass

required arguments: 'Length' 'BendingAngle' 'EntranceAngle' 'ExitAngle' 'PolynomB' 'MaxOrder' 'NumIntSteps' 'Energy'
optional arguments: 'FullGap' 'FringeInt1' 'FringeInt2' 'H1' 'H2' 'T1' 'T2' 'R1' 'R2'

12.20 BndMPoleSymplectic4Pass

required arguments: 'Length' 'BendingAngle' 'EntranceAngle' 'ExitAngle' 'PolynomA' 'PolynomB' 'MaxOrder' 'NumIntSteps'
optional arguments: 'FullGap' 'FringeInt1' 'FringeInt2' 'T1' 'T2' 'R1' 'R2'

12.21 BndMPoleSymplectic4RadPass

required arguments: 'Length' 'BendingAngle' 'EntranceAngle' 'ExitAngle' 'PolynomA' 'PolynomB' 'MaxOrder' 'NumIntSteps' 'Energy'
optional arguments: 'FullGap' 'FringeInt1' 'FringeInt2' 'T1' 'T2' 'R1' 'R2'

12.22 BendLinearPass

required arguments: 'Length' 'BendingAngle' 'EntranceAngle' 'ExitAngle'
optional arguments: 'K' 'ByError' 'FullGap' 'FringeInt1' 'FringeInt2' 'T1' 'T2' 'R1' 'R2'

12.23 CavityPass

required arguments:
'Length' 'Voltage' 'Energy' 'Frequency' optional arguments:
'TimeLag'

12.24 CorrectorPass

required arguments:
'Length' 'KickAngle' optional arguments: none

12.25 DriftPass

a =
'Length'
b =

12.26 QuadLinearPass

a =
'Length' 'K'
b =

'T1' 'T2' 'R1' 'R2'

12.27 QuadMPoleFringePass

a =
'Length' 'PolynomA' 'PolynomB' 'MaxOrder' 'NumIntSteps'
b =
'T1' 'T2' 'R1' 'R2'

12.28 StrMPoleSymplectic4Pass

a =
'Length' 'PolynomA' 'PolynomB' 'MaxOrder' 'NumIntSteps'
b =
'T1' 'T2' 'R1' 'R2'

12.29 StrMPoleSymplectic4RadPass

a =
'Length' 'PolynomA' 'PolynomB' 'MaxOrder' 'NumIntSteps' 'Energy'
b =
'T1' 'T2' 'R1' 'R2'

12.30 ThinMPolePass

required arguments: 'PolynomA' 'PolynomB' 'MaxOrder'
optional arguments: 'BendingAngle' 'T1' 'T2' 'R1' 'R2'

12.31 WiggLinearPass

required arguments: 'Length' 'InvRho'
optional arguments: 'KxKz' 'T1' 'T2' 'R1' 'R2'

12.32 IDTablePass

required arguments: 'Length' 'xkick' 'ykick' 'xtable' 'ytable' 'Nslice'
optional arguments: 'xkick1' 'ykick1' 'T1' 'T2' 'R1' 'R2'

References

[1] A. Terebilo *Accelerator Toolbox for Matlab*, SLAC-PUB 8732 (May 2001)