Dynamic_Aperture Program

David Sagan
June 21, 2022

Contents

1 Introduction

The *dynamic_aperture* program is for measuring the dynamic aperture. The concept of *dynamic aperture* is that a particle reaching a certain amplitude will quickly be resonantly driven to large amplitude where it is lost. This amplitude where the particle becomes unstable is the dynamic aperture. This is to be contrasted by the *physical aperture* which is the aperture where a particle strikes the wall of the beam chamber. The general idea in designing lattices is to make sure that the dynamic aperture is large enough so that, in the normal course of events, particles in the beam have a very small probability of getting lost due to their amplitude exceeding the dynamic aperture. For long term stability, a common rule of thumb is to design lattices such that the dynamic aperture is 10 times the beam sigma.[1] For injection studies, the minimum dynamic aperture will be determined in part by the size of the injected beam. In any case, if the dynamic aperture is larger than the physical aperture, increasing the dynamic aperture further will not help beam stability.

[1] While this might seem excessive, this rule of thumb gives some safety margin which is desirable since designs are never exact.

If there are no apertures set in lattice used by the *dynamic_aperture* program, the calculated aperture will be the dynamic aperture. If apertures are set in the lattice, the calculated aperture will be the minimum of the dynamic and physical apertures.

The *dynamic_aperture* program is built atop the Bmad software toolkit [1]. The Bmad toolkit is a library, developed at Cornell, for the modeling relativistic charged particles in storage rings and Linacs, as well as modeling photons in x-ray beam lines.

For historical reasons, the *Tao* program (another Bmad based program) is also capable of calculating the dynamic aperture. In fact both programs use the same underlying code for the tracking and analysis. Currently, the basic difference is that the *dynamic_aperture* program can handle *ramper* elements while *Tao* is not set up for this.

2 Running the Dynamic Aperture Program

The *dynamic_aperture* program comes with the "Bmad Distribution" which is a package which contains Bmad toolkit library along with a number of Bmad based programs. See the Bmad web site for more details.

If the Bmad Distribution is compiled with *OpenMP* enabled (see the documentation on the Bmad Distribution "Off-Site" setup for more details), the *dynamic_aperture* program can be run parallel. With OpenMP the computation load is distributed over a number of cores on the machine you are using. To set the number of cores set the *OMP_NUM_THREADS* environment variable. Example:

```
export OMP_NUM_THREADS=8
```

And run the program as normal as detailed below.

See the documentation for setting up the Bmad environment variables at

```
https://wiki.classe.cornell.edu/ACC/ACL/RunningPrograms
```

Once the Bmad environment variables have been set, the syntax for invoking the program is:

```
dynamic_aperture {<master_input_file_name>}
```

Example:

```
dynamic_aperture my_input_file.init
```

The *<master_input_file_name>* optional argument is used to set the master input file name. The default value is "*dynamic_aperture.init*". The syntax of the master input file is explained in §5.

Example input files are in the directory (relative to the root of a Distribution):

```
bsim/dynamic_aperture/example
```

3 Time Ramping — Time Varying Element Parameters

"*Ramping*" is the situation where lattice parameters are changing as a function of time over many turns. Ramping examples include changing magnet and RF strengths to ramp the beam energy or changing magnet strengths to squeeze beta at the interaction pont of a colliding beam machine.

Ramping is accomplished by defining *ramper* elements in the lattice file and setting *ramping_on* to True in the master input file (§5). Ramper elements will be applied to each lattice element in turn before particles are tracked through them. See the Bmad manual for documentation on *ramper* syntax.

Example:

```
ramp_e: ramper = \{*[e_tot]:\{4e+08, 4.00532e+08, 4.01982e+08, ...\}\},
              var = \{time\}, x_knot = \{0, 0.001, 0.002, ...\}

amp = 1e9;  omega = 0.167;  t0 = 0.053
ramp_rf: ramper = \{rfcavity::*[voltage]:amp*sin(omega *(time + t0)),
      rfcavity::*[phi0]:0.00158*time^2 + 2*q \}, var = \{time, q\}
```

The "**[e_tot]*" construct in the definition of *ramp_e* means that the ramper will be applied all elements (since the wild card character "*" will match to any element name), and it is the element's *e_tot* attribute (the element's reference energy) that will be varied.

In the above example, the *ramp_rf* ramper will be applied to all *rfcavity* elements with the cavity voltage and phase (*phi0*) being varied.

Important restriction: Only those ramper elements that have *time* as the first variable will be used and it will be this variable that is varied over time.

In the case where the reference energy *e_tot* or reference momentum *p0c* is being varied, the effect on an element will depend upon the setting of the element's *field_master* parameter. For example:

```
q1: quadrupole, k1 = 0.3
q2: quadrupole, k1 = 0.3, field_master = T
```

In this example, *q1* will have its *field_master* parameter set to *False* since the quadrupole strength was specified using the normalized strength *k1*. With *q1*, since *field_master* is False, varying the reference energy or momentum will result in the normalized strength *k1* remaining fixed and the unnormalized strength *B1_gradient* varying in proportion to the reference momentum. With *q2*, since *field_master* is True, the unnormalized strength *B1_gradient* will remain fixed and normalized *k1* will vary inversely with the reference momentum.

Before a simulation, individual ramper elements may be toggled on or off by setting the element's *is_on* attribute in the lattice file:

```
ramp_rf: ramper = ...   ! Ramper element defined.
ramp_rf[is_on] = F      ! Ramper element turned off.
```

4 Fortran Namelist Input

Fortran namelist syntax is used for parameter input in the master input file. The general form of a namelist is

```
&<namelist_name>
  <var1> = ...
  <var2> = ...
  ...
/
```

The tag *"&<namelist_name>"* starts the namelist where *<namelist_name>* is the name of the namelist. The namelist ends with the slash *"/"* tag. Anything outside of this is ignored. Within the namelist, anything after an exclamation mark *"!"* is ignored including the exclamation mark. *<var1>*, *<var2>*, etc. are variable names. Example:

```
&place
  section = 0.0, "arc_std", "elliptical", 0.045, 0.025
/
```

here *place* is the namelist name and *section* is a variable name. Notice that here *section* is a "structure" which has five components – a real number, followed by two strings, followed by two real numbers.

Everything is case insensitive except for quoted strings.

Logical values are specified by *True* or *False* or can be abbreviated *T* or *F*. Avoid using the dots (periods) that one needs in Fortran code.

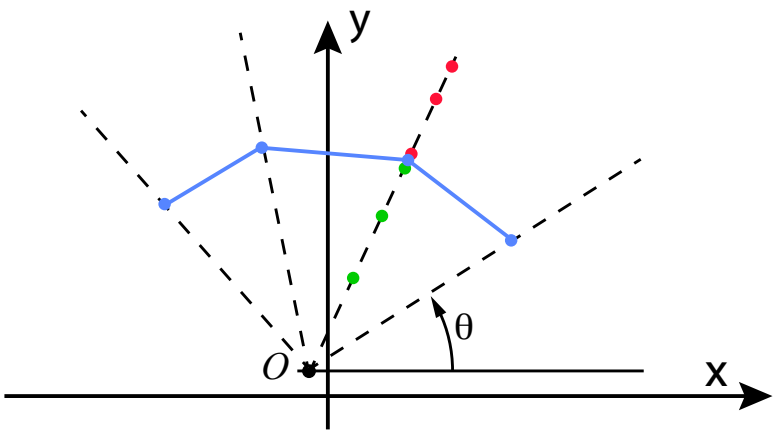

Figure 1: The calculation of a dynamic aperture curve in the x-y plane at a given initial p_z value involves calculating aperture curve points (blue dots) along a set of "rays" (dashed lines) having a common origin point ($\mathcal{O}$) which is taken to be the reference orbit. The line segments between points is simply for visualization purposes. The calculation of an aperture curve point along a given ray involves iteratively tracking particles with different starting (x, y) position values to find the boundary between stable (green dots) and unstable (red dots) motion.

5 Master Input File

The *master input file* holds the parameters needed for running the *dynamic_aperture* program. The master input file must contain a single namelist (§4) named *params*. Example:

```
&params
  lat_file    =  "lat.bmad"      ! Bmad lattice file
  dat_file = "da.dat"
  ramping_on = False
  ramping_start_time = 0
  set_rf_off = False
  dpz = 0.000, 0.005, 0.010
  bmad_com%radiation_damping_on = F
  da_param%min_angle = 0
  da_param%max_angle = 3.1415926
  da_param%n_angle = 0
  da_param%n_turn = 2000
  da_param%x_init = 1e-3
  da_param%y_init = 1e-3
  da_param%rel_accuracy = 1e-2
  da_param%abs_accuracy = 1e-5
  da_param%start_ele = ''
/
```

Parameters in the master input file are:

bmad_com%...

The *bmad_com* structure contains various parameters that affect tracking. For example, whether radiation damping is included in tracking. A full list of *bmad_com* parameters is detailed in the Bmad reference manual. Note: *bmad_com* parameters can be set in the Bmad lattice file as well. *Bmad_com* parameter set in the master input file will take precedence over parameters set in the lattice file.

da_param%n_turn

Number of turns to track.

da_param%start_ele

This parameter sets the starting element for tracking. If not set, the beginning element of the root branch is used. *da_param%start_ele* may be set to either the element name or element index.

da_param%n_angle

The number of boundary points calculated for a scan is set by the *da_param%n_angle* parameter.

da_param%min_angle, da_param%max_angle

These parameters set the ray minimum and maximum angles, labeled θ in Fig 1, in a scan. In the example above the angle ranges from 0 to pi. That is, the upper half-plane. These are typical settings since typically storage rings are vertically symmetric so the aperture curves should vertically symmetric as well.

The angles between adjacent rays is not uniform but are rather calculated to give a roughly equal spacing between boundary points. This is done by looking at the aperture points on a horizontal and a vertical ray and then scaling the ray angles appropriately).

da_param%rel_accuracy, da_param%abs_accuracy

These parameters set the relative and absolute accuracies that determine when the search for a boundary point is considered accurate enough.

If $r = \sqrt{(x - x_0)^2 + (y - y_0)^2}$ is the distance along any ray of the computed boundary point, where (x_0, y_0) are the coordinates of the origin point, the search for the boundary point will stop then the accuracy of the boundary point is below the desired accuracy σ_{cut} which is computed from

$$\sigma_{cut} = \sigma_a + r\,\sigma_r \tag{1}$$

with σ_a begin the absolute accuracy and σ_r being the relative accuracy.

da_param%x_init, da_param%y_init

These parameters set the initial x and y values used in the first two boundary point searches. The values of these parameters will not affect significantly affect the computed curve but will affect the computation time. If not set, these parameters will default to 0.001 meter.

dat_file

Name of the data output file. This name is required.

dpz

The *dpz* parameter array is a list of p_z values to use. The number of scans (dynamic aperture curves) that are produced is equal to the number of *pz* values.

ele_start

Name or element index of the element to start the tracking. Examples:

```
ele_start = "Q3##2"    ! 2nd element named Q3 in the lattice.
ele_start = 37         ! 37th element in the lattice.
```

The default is to start at the beginning of the lattice. Notice that the tracking starts at the downstream end of the element so the first element tracked through is the element after the chosen one.

lat_file

Name of the Bmad lattice file to use. This name is required.

ramping_on

If set to True, *ramper* control elements will be use to modify the lattice during tracking (§3). Default is False.

ramping_start_time

The starting (offset) time used to set *ramper* elements. This enables simulations to start in the middle of a ramp cycle. Default is 0.

set_rf_off

If set to *True*, the voltage on all RF cavity elements will be turned off. Default is *False*.

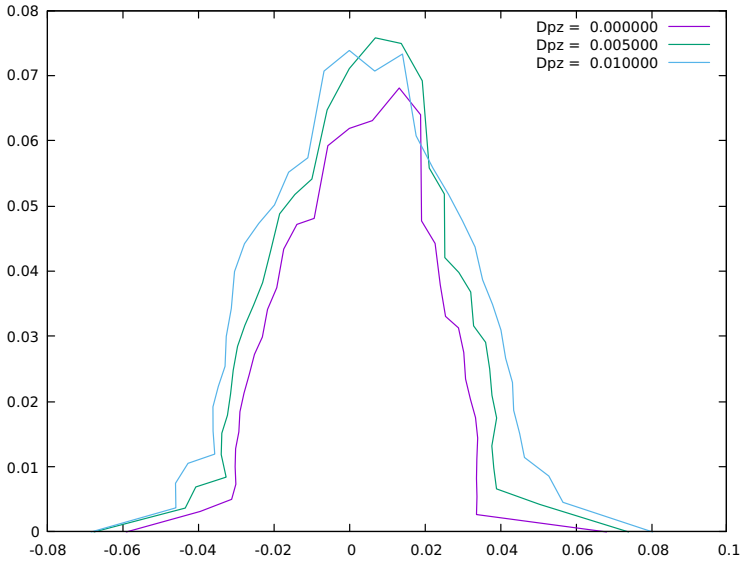

Figure 2: Example dynamic aperture plot using *gnuplot*.

6 Data Output and Plotting

The data output file whose name is set by *dat_file* will look like:

```
# lat_file              = chess_arc_pretzel_20150106.lat
# set_rf_off            = T
# da_param%min_angle    =        0.0
# da_param%max_angle    =        3.1
# da_param%rel_accuracy =   1.00E-02
... etc. ...
# da_param%n_angle      = 37
# gnuplot plotting command:
#   plot for [IDX=1:3] "da.dat" index (IDX-1) u 1:2 w lines ...

"dpz =  0.000000"
"x_ref_orb =  0.000124"
"y_ref_orb =  0.000037"
   0.068125   0.000000    544       Hit +X Side         Q11E
   0.033668   0.002675    442       Hit -Y Side         Q06E
   0.033807   0.005414    717       Hit +Y Side         Q26W
   0.033673   0.008195    410       Hit +X Side         B28W
   0.033759   0.011160    119       Hit -X Side         SEX_08E
   0.034006   0.014403    855       Hit +Y Side         SEX_21W
... etc. ...

dpz =  0.005000"
```

```
"x_ref_orb =  0.006591"
"y_ref_orb =  0.006591"
   0.073979   0.000000    904       Hit -Y Side        SEX_16W
   0.050379   0.004234    554       Hit -Y Side        SEX_19E
   0.038989   0.006603    987       Hit -X Side        SEX_39E
   0.038242   0.009842    447       Hit +Y Side        SEX_15W
   0.037746   0.013196    365       Hit +X Side        SEX_41E
... etc. ...
```

The top part of the data file will be a record of input parameter values. This is followed by a number of data blocks, one for each setting of *dpz*. The five columns of these data blocks are:

```
1 & 2: x_aperture, y_aperture
3:     Number of turns a particle initially at the aperture limit survived.
4:     Transverse location where particle died.
5:     Lattice element where particle died.
```

Note that a particle will "die" if it hits an aperture or its amplitude is beyond the setting of *bmad_com%max_aperture_limit*. The default value of this maximum aperture is 1000 meters.

One way to plot the data is to use the *gnuplot* program (documentation for gnuplot is available using a web search). Run *gnuplot* and use the command printed in the top section of the data file. An example of what such a plot looks like is shown in Fig. 2.

References

[1] D. Sagan, "Bmad: A Relativistic Charged Particle Simulation Library" Nuc. Instrum. & Methods Phys. Res. A, **558**, pp 356-59 (2006).