Long Term Tracking Program

David Sagan
August 19, 2022

Contents

1 Introduction

The *long_term_tracking* program is for long term (over many turns) tracking of a particle or a beam in a ring. The *long_term_tracking* program can track spin as well as simulate such things as element misalignments, wake fields, higher order mode cavity resonances, energy ramping, beta squeezing, etc. The output of the program will be such things as turn-by-turn particle tracks or beam statistics including beam sizes and polarizations.

The *long_term_tracking* 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.

2 Running the Long Term Tracking Program

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

See the documentation for setting up the Bmad environment variables at

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

The *long_term_tracking* program comes in two versions:

```
long_term_tracking       ! Single threaded version
long_term_tracking_mpi   ! Multi-threaded version
```

The first is a single threaded version and the second is a multi-threaded version using MPI.[1]

Once the Bmad environment variables have been set, the syntax for invoking the single threaded version of the *long_term_tracking* program is:

```
long_term_tracking {<master_input_file_name>}
```

Example:

```
long_term_tracking my_input_file.init
```

[1] Note to Distribution maintainers: The single threaded version will be built when compiling a Bmad Distribution. If MPI is not enabled (which is the default setting), the MPI version will not be built. The MPI version can be built by setting *export ACC_ENABLE_MPI=Y* and then using the "*mk*" command in the *bsim* directory to build the executable. See the documentation on the Bmad web site for more details.

The *<master_input_file_name>* optional argument is used to set the master input file name. The default value is "*long_term_tracking.init*". The syntax of the master input file is explained in §12.

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

```
bsim/long_term_tracking/example
```

When running the MPI version threaded over multiple computers, the details of how to start the process will vary depending upon the installation. See your local Guru for details. When running on a single machine, the typical command will look like

```
mpiexec -n <num_processes> long_term_tracking_mpi {<master_input_file_name>}
```

where *<num_processes>* is the number of processes.

3 Definition of Some Terms

Beam

A *beam* consists of a train *bunches*. Each *bunch* consists of a number of particles contained within one RF bucket.

Radiation Damping and fluctuations

At a given particle position, the energy kick dE that a particle receives due to emission of radiation can be decomposed into two parts. One part is the average kick dE_d and the other part is a random fluctuation dE_r around the average

$$dE = dE_d + dE_r \tag{1}$$

An analysis shows that the average part dE_d leads to damping with respect to the closed orbit and the fluctuation part leads to excitation of the particle motion. Over many turns, there is an equilibrium reached which gives a particle beam its natural size. While it is not physically possible to have radiation damping without excitation and vice versa, this can be done with ease in a simulation.

PTC

PTC is a library developed by Étienne Forest to handle Taylor maps to any arbitrary order. In particular, PTC is used for constructing maps for the *long_term_tracking* program. PTC can also be used to do element-by-element tracking.

4 Simulation Modes

There are a number of simulation "modes" which determine what is done by the program. The simulation mode is set by the *ltt%simulation_mode* parameter in the master input file (§12). Possible settings of *ltt%simulation_mode* are:

```
"CHECK"    ! Quick tracking check.
"SINGLE"   ! Single particle tracking.
"BEAM"     ! Beam tracking.
"STAT"     ! Lattice statistics.
```

"CHECK"
In this mode, a particle is tracked from *ltt%ele_start* to *ltt%ele_stop* three times using a different tracking method each time. The three tracking methods are 1) using a set of maps (§6), and element-by-element tracking using 2) *Bmad* and 3) *PTC*. The results are then printed to the terminal. This mode can be used to get a sense of how the three different methods compare to each other. The starting coordinates are set by *particle_start* commands in the lattice file. Included in the output is the transfer matrix (Jacobian) with respect to the tracked orbit for each of the three tracking methods. These transfer matrices are computed using finite differences when the starting position is varied. The variation used in the computation can be specified by setting in the master input file the vector *bmad_com%d_orb*. This vector has 6 components, one for each phase space component. That is, *bmad_com%d_orb(1)* sets the variation of the x phase space component, etc.

No data files are produced in this mode.

Radiation fluctuations are turned off in this mode but radiation damping will still be applied if turned on.

"SINGLE"
In this mode a single particle is tracked for *ltt%n_turns* turns. The starting coordinates are set by *particle_start* commands in the lattice file.

The name of the data file is set by the *ltt%phase_space_output_file* parameter (§12). The particle position will be output every *ltt%particle_output_every_n_turns* turns. If this parameter is zero or negative, the position will be outputted at every element.

"BEAM"
In this mode a particle beam is tracked for *ltt%n_turns* turns. The output data file(s) format is described in Section §16.

"STAT"
In this mode statistics about the lattice are calculated. No long term tracking is done.

Three data files are produced:

```
twiss.dat          ! Twiss parameters
coupling.dat       ! Coupling parameters
closed_orbit.dat   ! Closed orbit
```

5 Tracking Methods

When tracking with *ltt%simulation_mode* set to *"SINGLE"* or *"BEAM"*, the setting of the parameter *ltt%tracking_method* determines how particles are tracked. Possible settings are:

```
"MAP"      ! Tracking using maps.
"PTC"      ! Element-by-element tracking with PTC. Slow.
"BMAD"     ! Element-by-element tracking using Bmad. Default. Slow.
```

which is used to determine if tracking is done using a map or not. If a map is used, the order of the map (the degree at which the Taylor series comprising the map are truncated) is given by *ltt%map_order* parameter. A larger map order will mean better accuracy at the expense of more computation time. Tracking using a map will be extremely quick compared to element-by-element tracking. However, map tracking can be inaccurate if the map order is too small or the particle amplitude is too large.

Only the *BMAD* method is able to handle a machine that is being ramped (§9). That is, when *ltt%ramping_on* set to True. The program will detect if there is a conflict and issue an error message and stop.

6 Map Tracking

The *long_term_tracking* program uses the PTC/FPP library of Étienne Forest to handle Taylor maps which can be constructed to any arbitrary order. Maps will transport both orbital and spin coordinates. In particular, PTC is used for constructing the map(s) used for tracking when the *ltt%tracking_method* parameter is set to "*MAP*" (§12.3). PTC tracking is also used when *ltt%tracking_method* is set to "*PTC*". In this case, tracking is done element-by-element using symplectic integration.

Note: Using maps is not compatible when a machine is being ramped (§9).

Sometimes it is convenient to exclude certain lattice elements from a map. For example, the beam-beam interaction is highly non-linear so excluding any *beambeam* elements can improve map accuracy at larger amplitudes. Which elements are excluded is determined by the setting of *ltt%exclude_from_maps* which is a list of what elements are to be excluded. Elements can be excluded for a number of reasons. When an element is excluded, multiple maps are created, one for each of the sections of the lattice where there are no excluded elements. In this case, tracking consists of using a map to track from one excluded element to the next followed by tracking through the excluded element.

Important! Maps cannot model RF elements that have a frequency not commensurate with the one-turn frequency. The reason for this is that a map is just a function that maps the particle's starting phase space coordinates to the particle's ending coordinates independent of how many turns the particle has been tracked. But if an RF element has a frequency not commensurate with the one-turn frequency the kick given a particle going through the element will depend upon the turn index. Maps can be used to model a lattice with such RF elements, but in this case such elements must be excluded from be incorporated in any map by setting *ltt%exclude_from_maps* appropriately.

Depending upon how things are setup, a *PTC* map can include radiation damping and fluctua-

tions effects. [2] Damping and excitation are controlled by two parameters that can be set in the master input file:

```
bmad_com%radiation_damping_on
bmad_com%radiation_fluctuations_on
```

The problem with trying to use a map to track a particle's spin when there are radiation effects present is that a map may not properly model the effect of the radiation fluctuations on the spin precession (radiation damping is not problematic in itself but that is small comfort). To get around this, the switch *ltt%split_bends_for_stochastic_rad* may be set True. In this case, all bend elements are split in the center and special radiation marker elements are placed in between the split bends. These radiation marker elements are excluded when constructing maps so the result is many maps whose boundaries will be at the bend centers. When tracking, the maps will not incorporate stochastic radiation effects (but will still have radiation damping) and a stochastic radiation kick will be put in at the radiation marker elements as well as any other elements that are excluded from the maps.

This technique of bend splitting for the purposes of including radiation effects was originally implemented in a program called *SLICK* and later in the offshoot program *SLICKTRACK*. As such, this algorithm is known as *SLICK* tracking. A drawback with *SLICK* tracking is the number of maps needed to track a particle over one turn may be considerably more than just using one map (essentially there will be one map for every bend in the lattice). This will lengthen computation times. This being so, it is recommended that *SLICK* tracking only be used when needed. That is, when both radiation effects and spin tracking are to be simulated. *SLICK* tracking is also potentially problematical in that radiation effects are not taken into account for non-dipole elements that are included in any map. If there are any such elements (for example, wiggler elements), these elements should be excluded from being included in any map.

Maps are saved to a file for use if the program is rerun. For linear maps, an ASCII file can be produced by setting *ltt%map_ascii_output_file*.

7 Tuning Map and PTC Tracking Parameters

When tracking using maps or element-by-element with PTC there are a few points to keep in mind. First is that *PTC* tracks through a lattice element step by step. This is true for both map creation and symplectic integration. This means that the setting of the element parameter *integration_order*[3] or *num_steps* (or *ds_step*) for each element will affect the accuracy and speed of the computations. Bmad tries to choose reasonable default settings for the integration order and number of steps however the calculation is not perfect. To make sure that the integration order and number of steps is set properly, vary both and choose values (which can be different for different elements) such that the number of steps and integration order is minimal (to minimize computation time) while at the same time is large enough so that results do not change

[2] Radiation fluctuations are included by actually using two maps to track a particle from one point to the next. One map represents the transport with damping and the second map represents the fluctuations. When a particle is tracked, the first map with damping is applied and then the second map is applied using six random numbers.

[3] Note: Valid settings for *integration_order* are 2, 4, and 6

significantly if the number of steps or is varied. Generally it is much better to use a large integration order and a small step size rather than vice versa with the proviso that for elements with a longitudinally varying field (think wigglers or undulators), the step size must be small compared to the typical longitudinal length scale over which the field is varying (this length scale is the pole period length with with wigglers and undulators).

Another thing to keep in mind is that whether a map will give accurate results is dependent on a number of factors. One factor is the order of the map. Generally higher order is better but will take more computation time. When the lattice is tracked using a single map, the tracking is only valid when the tracked particles are far from any strong resonances. That is, if you are interested in tracking halo particles, you probably should not be using a single map.

In terms of speed, using maps will be the fastest, using standard Bmad tracking will be much slower, and using PTC element-by-element tracking will be slowest.

In terms of symplecticity, both the PTC tracking and map tracking will be symplectic. Bmad is not symplectic but the deviation from symplecticity is generally fairly small. If the radiation effects are large enough, the radiative stochastic noise will negate any non-symplectic effects and standard Bmad tracking can be used. A very rough rule of thumb is that if the damping times or the number of turns tracked are under 100,000 turns then Bmad standard tracking can be used.

PTC element-by-element tracking cannot be done when using the MPI version of the *long_term_tracking* program.

8 Correcting the Orbit when Radiation is Present

In storage rings, When radiation damping is simulated, the orbit that is flat when there is no radiation will show a “sawtooth" pattern” in a plot of beam energy versus position. This will lead to a nonzero orbit. In an actual ring, the non-zero orbit will be compensated using steerings. Thus, to simulate the actual ring, compensating steerings should be added to the simulated lattice as well. How to do this is covered in the Bmad and Tao Cookbook available from the Bmad web site.

9 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. These ramper elements will be applied to each lattice element in turn before particles are tracked through the element. 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 *rıcavity* 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.

There are two modes that can be selected to determine how ramper elements are applied. If *ltt%ramp_update_each_particle* is set to True (the default), rampers are applied to a given lattice element as each particle of the beam passes through the element using the time the particle would pass through the center of the element. If set to False, rampers are applied to a given element only once per beam passage using the time of the center of the bunch passing the center of the element. This will be less accurate but will speed up the calculation. Implicit in this is the assumption that controlled parameters are varying slow enough so that the rampers only need be applied once per beam passage. This is done to minimize computational time. If ramper controlled parameters are slowly varying, for example if ramping magnet strengths with energy ramping which occurs over many turns, calculating once per beam passage should be a good approximation.

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.
```

If *ltt%ramping_on* is set to True, ramping will be done if *ltt%simulation_mode* is set to "*BEAM*" or "*SINGLE*". Ramping will be ignored in the other modes.

long_term_tracking program parameters that affect ramping in the simulation are:

```
ltt%ramping_on
ltt%ramping_start_time
```

When ramping, the *ltt%simulation_mode* (§5) must be set to "*BMAD*". The program will detect if there is a conflict and issue an error message and stop.

10 Initial Particle Positions

When the *ltt%simulation_mode* (§4) is set to *"SINGLE"* or *"CHECK"*, the initial particle position will be set to the initial beam center position. The parameters that are used to determine the beam center position are

```
beam_init%center           ! (x, px, y, py, z, pz)
beam_init%spin             ! (Sx, Sy, Sz)
beam_init%use_particle_start
ltt%add_closed_orbit_to_init_position
```

If *beam_init%use_particle_start* is set to *True*, instead of using the parameters *beam_init%center* and *beam_init%spin*, the *particle_start* parameters that are set in the lattice file will be used.

If the *ltt%add_closed_orbit_to_init_position* logical is set to *True*, the closed orbit position is added in to the center position calculation.

When the *ltt%simulation_mode* is set to *"BEAM"*, the initial particle positions are calculated using the *beam_init* structure as discussed in the *Beam Initialization* chapter of the Bmad manual. Like the other modes, if the *ltt%add_closed_orbit_to_init_position* logical is set to *True*, the closed orbit position is added to the initial particle positions. To read in a file with beam particle positions, set the *beam_init* structure appropriately. Example:

```
beam_init%file_name = 'beam_particle_file.init'
```

In all cases, tracking will start at the lattice element set by *ltt%ele_start*.

The *beam_init* structure:

```
type beam_init_struct
  character(200) :: position_file = ''       ! Initialization file name.
  character distribution_type(3)             ! "ELLIPSE", "KV", "GRID", "".
  type (ellipse_beam_init_struct) ellipse(3) ! For ellipse beam distribution
  type (kv_beam_init_struct) KV              ! For KV beam distribution
  type (grid_beam_init_struct) grid(3)       ! For grid beam distribution
  logical use_particle_start = F     ! Use particle_start?
  character random_engine            ! "pseudo" (default) or "quasi".
```

```
  character random_gauss_converter   ! "exact" (default) or "quick".
  real center(6) = 0                 ! Bench center offset.
  real center_jitter(6) = 0          ! Bunch center rms jitter
  real emit_jitter(2)   = 0          ! %RMS a and b-mode emittance jitter
  real sig_z_jitter     = 0          ! bunch length RMS jitter
  real sig_pz_jitter     = 0         ! pz energy spread RMS jitter
  real random_sigma_cutoff = -1      ! -1 => no cutoff used.
  integer n_particle = 0             ! Num of simulated particles per bunch.
  logical renorm_center = T          ! Renormalize centroid?
  logical renorm_sigma = T           ! Renormalize sigma?
  real(rp) spin(3) = 0, 0, 0         ! Spin (x, y, z)
  real a_norm_emit = 0               ! a-mode normalized emittance
  real b_norm_emit = 0               ! b-mode normalized emittance
  real a_emit = 0                    ! a-mode emittance
  real b_emit = 0                    ! b-mode emittance
  real dpz_dz = 0                    ! pz vs z correlation.
  real dt_bunch = 0                  ! Time between bunches.
  real sig_z = 0                     ! Z sigma in m.
  real sig_pz = 0                    ! pz sigma.
  real bunch_charge = 0              ! Charge in a bunch.
  integer n_bunch = 0                ! Number of bunches.
  character species = ""             ! Species to track.
  logical full_6D_coupling_calc = F  ! Use 6x6 1-turn mat to match
                                     !              the distribution?
  logical use_t_coords = F  ! If true, the distributions will be
                            !   calculated using time coordinates
  logical use_z_as_t   = F  ! Only used if use_t_coords = T:
                            !   True:  The z coordinate stores the time.
                            !   False: The z coordinate stores the s-position.
end type
```

11 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.

12 Master Input File

The *master input file* holds the parameters needed for running the long term tracking program. The master input file must contain a single namelist (§11) named *params*. Example:

```
&params
  ! Input
  ltt%lat_file    =  "lat.bmad"      ! Bmad lattice file
  ltt%ramping_on = False
  ltt%ramp_update_each_particle = True
  ltt%ramping_start_time = 0

  ! Output parameters
  ltt%phase_space_output_file = ""
  ltt%action_angle_output_file = ""
  ltt%beam_binary_output_file = "beam.dat"
  ltt%averages_output_file = "average.dat"
  ltt%custom_output_file = ""
  ltt%averages_output_every_n_turns = 200
  ltt%particle_output_every_n_turns = 200
  ltt%only_live_particles_out = T
  ltt%core_emit_cutoff    = 0.2, 0.5
  ltt%core_emit_combined_calc = T
  ltt%print_on_dead_loss = -1
  ltt%map_file_prefix = "ltt"
  ltt%map_ascii_output_file = ""

  ltt%column(1) = "n_turn", "N_turn", "i7"
  ltt%column(2) = "rf0##1[voltage]", "Volt", "f10.0"
  ltt%column(3) = "rf0##1[phi0] + z/rf0##1[rf_wavelength]", "dPhi", "f12.8"

  ! Simulation parameters
  ltt%simulation_mode     = "BEAM"
  ltt%tracking_method     = "BMAD"
  ltt%ele_start           = ""
```

```
  ltt%ele_stop           = ""            ! Used for "CHECK" simulation_mode
  ltt%n_turns            = 1000
  ltt%map_order          = 5
  ltt%rfcavity_on        = True
  ltt%timer_print_dtime  = 300
  ltt%random_seed        = 0
  ltt%ptc_aperture       = 0.1, 0.1
  ltt%exclude_from_maps  = "beambeam::*"
  ltt%split_bends_for_stochastic_rad = False
  ltt%symplectic_map_tracking = False
  ltt%dead_cutoff        = 0.99
  ltt%b_emittance        = 1e-9          ! Used with space charge calc.
  ltt%debug              = F
 bmad_com%spin_tracking_on = T          ! See Bmad manual for
 bmad_com%radiation_damping_on = F      !    bmad_com parameters.
 bmad_com%radiation_fluctuations_on = F

 ! Beam initialization
 ltt%add_closed_orbit_to_init_position = True
 beam_init%use_particle_start = F
 beam_init%center = 0, 0, 0, 0, 0, 0
 beam_init%n_particle = 10
 beam_init%spin = 0, 0, 0        ! See Bmad manual for complete list
 beam_init%a_emit = 1e-8         !    of beam_init_struct parameters.
 beam_init%b_emit = 1e-8
 beam_init%sig_z = 1e-4
 beam_init%sig_pz = 1e-4
/
```

The namelist parameters can be divided into three categories: Input files, output parameters, and simulation parameters.

12.1 Input Files

The input files that can be specified in the *params* namelist of the master input file are:

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

12.2 Output Files

Parameters in the master input file that affect the output are:

ltt%action_angle_output_file
Used with *ltt%simulation_mode* set to *"BEAM"*. The name of the ASCII file or files that are

created to hold the particle positions in action-angle coordinates. A file is created every *ltt%particle_output_every_n_turns* turns. See Section §16 for more details.

If *ltt%action_angle_output_file* is blank, no particle output file is created. See also the parameter *ltt%beam_binary_output_file* which will produce a portable binary file. Also see *ltt%phase_space_output_file*.

ltt%averages_output_every_n_turns

Sets the number of turns between data rows of data files specified by:

```
ltt%averages_output_file
ltt%custom_output_file
```

If set to -1 (the default), the output will only happen at the last turn. If set to 0, output will happen on the beginning turn (turn 0) and the last turn. Example:

```
ltt%averages_output_every_n_turns = 1000
```

In this example, output will happen every 1000 turns. For particle phase space output, use the *particle_output_every_n_turns* parameter.

ltt%averages_output_file

Used with *ltt%simulation_mode* set to *"BEAM"*. There are three data files generated here. The reason to generate multiple data files is to avoid a file with extremely long lines. the names of the three files will be generated by appending the suffixes "*.ave*", "*.sigma*", and "*.emit*" to the name. If the *ltt%averages_output_file* string contains a hash "*#*" character, rather than appending the suffixes, the suffixes, minus the beginning dot character, will be substituted for the hash mark.

The *averages* file with the *.ave* suffix will contain the following columns:

```
Turn                          -- Turn number
N_live                        -- Number of live particles
Time                          -- Time from start
Polarization                  -- Spin polarization
<Sx>, <Sy>, <Sz>              -- Average spin components
<x>, <px>, <y>,
      <py>, <z>, <pz>         -- Beam centroid.
<p0c>                         -- Reference momentum
Sig_x, Sig_px, Sig_y,
      Sig_py, Sig_z, Sig_pz   -- Phase space sigmas
```

The *sigma* file with the *.sigma* suffix will contain the following columns:

```
Turn                          -- Turn number
N_live                        -- Number of live particles
Time                          -- Time from start
Sigma_Matrix                  -- Sigma matrix (21 columns)
```

The 6x6 sigma matrix Σ is a measure of the beam size defined by

$$\Sigma_{ij} = \langle dr_i \, dr_j \rangle \tag{2}$$

where $\mathbf{dr}$ is the deviation of the particle phase space position from the average and $< \ldots >$ denotes an average over all particles. Since the matrix is symmetric, only 21 columns are needed.

The *emittance* file with the *.emit* suffix will contain the following columns:

```
Turn                          -- Turn number
N_live                        -- Number of live particles
Time                          -- Time from start
emit_a, emit_b, emit_c        -- Normal mode emittances
kurtosis_x, kurtosis_y,
      kurtosis_z              -- Kurtosis in x, y, and z directions
skew_x, skew_y, skew_z        -- Skewness in x, y, and z directions
core_emit_a, core_emit_b,
      core_emit_c             -- "Core" emittance for
core_emit_a, core_emit_b,     --          ltt%core_emit_cutoff(1)
      core_emit_c             -- "Core" emittance for
core_emit_a,                  --          ltt%core_emit_cutoff(2)
... etc ...
```

Skewness κ_3 is defined by

$$\kappa_{3i} \equiv \frac{\langle dr_i^3 \rangle}{\sigma_i^3}, \quad i = x, y, \text{or}, z \tag{3}$$

And kurtosis κ_4 is covered in Section 15.

Each line in the file represents the averages at the end of a turn and a new line is added every *ltt%averages_output_every_n_turns* number of turns. If *ltt%averages_output_file* is blank, no averages file is produced. Data is always recorded when the beam is at the *ltt%ele_start* position independent of where the lattice begins and ends.

ltt%beam_binary_output_file

Used with *ltt%simulation_mode* set to *"BEAM"*. This parameter sets the name of the file that is created to hold the particle positions at the end of tracking. This is similar to *ltt%phase_space_output_file* but here the file will be a binary file that is portable to other programs and can also be used as input to the *long_term_tracking* program.

ltt%column(:)

A custom output file can be created that records beam and lattice parameters every N^{th} turn where N is set by *ltt%averages_output_every_n_turns*. The name of the file is set by *ltt%custom_output_file*.

Columns in the custom file are specified by setting *ltt%column(<i>)*. The general syntax is:

```
ltt%column(<i>) = "<expression>", "<header-name>",  "<format>"
```

where *<i>* is the column index, *<expression>* is an arithmetical expression that determines the values displayed in the column, *<header-name>* is a descriptive string used in constructing the header line (first line) in the file, and *<format>* is the format specifier used when converting the value of an expression into a string. Example:

```
ltt%column(1) = "n_turn", "N_turn", "i7"
ltt%column(2) = "rf0##1[voltage]", "Volt", "f10.0"
ltt%column(3) = "rf0##1[phi0] + z/rf0##1[rf_wavelength]", "dPhi", "es12.8"
```

Here three columns are specified. Parameters that are used in a column's *<expression>* can be element attributes or beam quantities. Using element attributes can be helpful when ramping is used. In the above example, the second column's expression is the *voltage* of the first element named *rf0* in the lattice (see the Bmad manual for lists of element parameters). Other parameters are:

```
n_turn     -- Turn number
time       -- Time at the N\Th turn averaged over the beam.
x, px, y, py, z, pz
           -- Particle phase space averaged over the beam.
```

Thus the third column in the above example is the averaged particle phase with respect to the zero crossing phase of the first element named *rf0* in the lattice. [Note that z is always evaluated at the start of the turn and not at the position of the RF cavity.]

ltt%map_ascii_output_file
For linear maps, An ASCII file of the maps can be produced by setting this parameter.

ltt%map_file_prefix
Prefix, including possible directory specification, for creating files in which maps are saved. Map files are used to quickly retrieve maps previously computed in prior running of the program. The default prefix is *"ltt"*. The general form of a map file name is

```
<map_file_prefix>.<hash>.map
```

where *<hash>* is a hash string that represents input parameters (like the order of the map) used to compute the map(s). This hash string is used so that map(s) constructed with one set of parameters are not used if the parameters are changed.

ltt%core_emit_combined_calc
Determines how core emittances are calculated. See §14 for more details. The default value is *True*.

ltt%core_emit_cutoff
The *ltt%core_emit_cutoff* parameter is an array of values used to calculate the "core" emittances (§14). Up to 40 values can be given. If a value is non-positive, that value and any value after that are ignored. The default first value is 0.5 and the reset default to negative. The value of *ltt%core_emit_cutoff* must be less than or equal to one. Using a value near zero may lead to invalid results if the number of particle to be retained is too small since there will be statistical fluctuations of order $1/\sqrt{N_c}$ where N_c is the number of particles in the core. The core emittance will be in the *.emit* output data file (see *ltt%averages_output_file*).

ltt%custom_output_file
A custom output file can be created that records beam and lattice parameters every N^{th}

turn where N is set by *ltt%averages_output_every_n_turns*. The columns of the file are set by the *ltt%column(:)* parameters.

ltt%only_live_particles_out
If this parameter is True (the default), the *ltt%particles_output_file* file will not record dead particles. If set to False, all particles will be recorded. Note: The last column of this file indicates if the particle is alive or dead.

ltt%particle_output_every_n_turns
Sets the number of turns between particle position data files. If set to -1 (the default), a file will only be generated at the last turn. If set to 0, a file will be generated at the beginning turn (turn 0) and at the last turn. Example:

```
ltt%particle_output_every_n_turns = 1000
```

In this example, output will happen every 1000 turns. For "averages" files the corresponding parameter is *averages_output_every_n_turns*.

ltt%phase_space_output_file
Used with *ltt%simulation_mode* set to *"BEAM"*. The name of the ASCII file or files that are created to hold the particle positions. A file is created every *ltt%particle_output_every_n_turns* turns. See Section §16 for more details. If *ltt%phase_space_output_file* is blank, no particle output file is created. See also *ltt%beam_binary_output_file* which will produce a portable binary file. Also see *ltt%action_angle_output_file*.

ltt%print_on_dead_loss
After each turn, the number of particles that are still living (have not hit an aperture) are counted and if the percentage of particles that have died, counting from the last time a message was printed, is larger than *ltt%print_on_dead_loss*, a message is printed. For example, if *ltt%print_on_dead_loss* is set to 0.01, every time the beam loses 1% of the particles a message is printed. Default is -1 which will cause a message printed every turn where there is particle loss.

12.3 Simulation Parameters

Parameters in the master input file that affect the simulation are:

ltt%add_closed_orbit_to_init_position
If set *True* (the default), initial particle positions are set equal to the input particle positions plus the closed orbit position. See Section §10.

beam_init%...
This sets the initial beam distribution. See Section §10.

If spin tracking is on (*bmad_com%spin_tracking_on* = T), and if all three components of *beam_init%spin* are zero, the closed orbit spin direction (n_0) is used.

Also: the values of *beam_init%a_emit* and *beam_init%b_emit* are used in the high energy space charge calculation (§13). If set to zero (the default) or negative, the emittance as calculated via radiation integrals will be used. Note: A value of

bmad_com%...
The *bmad_com* structure contains various parameters that affect tracking. For example, whether radiation damping and fluctuations are 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.

ltt%dead_cutoff
Sets the cutoff for the number of dead particles below which the simulation will stop. For example, if *ltt%dead_cutoff* is set to 0.01, when 1% of the beam has been lost, the simulation will stop. The default value for *ltt%dead_cutoff* is 0.99

ltt%debug
This parameter is used for debugging the program. This is not of interest to the general user.

ltt%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. Also see *ltt%ele_stop*.

ltt%ele_stop
Used when *ltt%simulation_mode* is set to *"CHECK"*. *ttl%ele_stop* sets the stopping point for tracking to be the downstream edge of the element. Also see *ltt%ele_start*. Default if not set or set to a blank string is for *ltt%ele_stop* to be equal to *tll%ele_start*.

ltt%exclude_from_maps
List of elements to exclude when constructing maps for *SLICK* tracking. These elements will be individually tracking. The default value is *"beambeam::*"* which excludes any *beambeam* element. See the Bmad manual section on "*Matching to Lattice Element Names*" for details on the format for such lists.

ltt%map_order
Map order. See Section §5. The default is what is set in the lattice file and if not set in the lattice file the default is 3. Note: *ltt%map_order* is only used when generating a map. When a map is read in from a file, the order of this map is independent of the current setting of *ltt%map_order*.

ltt%n_turns
Number of turns to track. See Section §5.

ltt%ptc_aperture
The PTC code does not have apertures. This being the case, for *ltt%tracking_method* set to *"MAP"* or *"PTC"*, *ltt%ptc_aperture*, which is a 2-vector, defines *x* and *y* apertures. The default is 0.1 meter in both the horizontal and vertical. When used, the aperture is applied at the beginning/end of the lattice. PTC has an internal aperture of 1.0 meter. To be safe, the *long_term_tracking* program will additionally impose a 0.9 meters aperture independent of the setting of *ltt%ptc_aperture*.

ltt%ramp_update_each_particle
Default is True. If set to false, to save time, only apply rampers to a given lattice element once per beam passage (§9).

ltt%ramping_on
If set to True, *ramper* control elements will be use to modify the lattice during tracking (§9). Default is False.

ltt%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.

ltt%random_seed
The random number seed used by the random number generator. If set to 0, the system clock will be used. That is, if set to 0, the output results will vary from run to run.

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

ltt%simulation_mode
Sets the simulation mode for the program. See Section §4 for more details.

ltt%split_bends_for_stochastic_rad
Use a stochastic radiation point in the middle of all the bends instead of including stochastic radiation effects in the transport maps (§6)? Default is False. For testing, can also be used with non-map tracking.

ltt%symplectic_map_tracking
If False (the default), the maps used for tracking will be a set of truncated Taylor series polynomials. If True, the tracking maps will be derived from the Taylor map by partially inverting it forming an implicit symplectic map. The advantage of the symplectic map is that it is symplectic. The disadvantage is that, being an implicit map, the computation time will be longer.

ltt%timer_print_dtime
The program will print a tracking status message every so often. The nominal time between status messages is set by *ltt%timer_print_dtime* which is a number in seconds.

ltt%tracking_method
String switch which sets how particles are tracked when the *simulation_mode* is set to "*BEAM*" or "*SINGLE*". Possible settings are:

```
"MAP"      ! Tracking using maps.
"PTC"      ! Element-by-element tracking with PTC. Slow.
"BMAD"     ! Element-by-element tracking using Bmad. Default. Slow.
```

See sections §6 and §5 for more details.

13 Simulations with Ultra-Relativistic Space Charge Effects

A space charge kick can be applied in the simulation. The space charge kick is calculated using an approximation suitable at ultra-relativistic energies. The kick is turned on by setting the logical *parameter[high_energy_space_charge_on]* to *True* (default is *False* in the lattice file. See the Bmad manual documentation for more details. The high energy space charge kick is only applied when *ltt%tracking_method* is set to *"BMAD"*.

The a and b mode emittances are used as input to the space charge calculation. [If there is no coupling, the a mode corresponds to the "horizontal" mode and the b mode corresponds to the "vertical" mode.] These emittance can be specified by setting *beam_init%a_emit* and/or *beam_init%b_emit*. If set to zero or negative, an emittance is calculated using a radiation integral calculation. Additionally, the longitudinal beam size is calculated via radiation integrals.

14 Core Emittance

The "*core*" emittance is defined by the *long_term_tracking* program to be the emittance as computed using some fraction of the beam set by *ltt%core_emit_cuttoff*. For example, if this parameter is set to 0.9, The 10% of a bunch with the largest amplitudes will be ignored and the core emittance is computed from the remaining 90%.

The core emittance is a way for characterizing the core of the beam. To characterize the tails there is kurtosis§15.

Other methods for computing a core emittance exist than what is implemented in the *long_term_tracking* program. For example, Gulliford et. al.[4] use the beam density at the beam center as a measure of core emittance. With realistic beam distributions, the algorithms here should give comparable results for *ltt%core_emit_cuttoff* C set low enough. The advantage of the algorithms here is that they are flexible and can include particles that are further away from the center. In any case, it should always be keep in mind that using only a few numbers, like emittance, core emittance, and kurtosis, to characterize the entire particle distribution, can potentially be misleading.

14.1 Combined Core Emit Calculation

If *ltt%core_emit_combined_calc* is set to True (the default), the computation proceeds as follows:

1. The beam size sigma matrix Σ_{ij} for the whole bunch is analyzed to extract the normal mode eigenvectors and emittances as as shown in a paper by Wolski[3].

2. The "6D amplitude" A of a particle is defined to be

$$A(J_a, J_b, J_c) \equiv \frac{J_a}{\varepsilon_a} + \frac{J_b}{\varepsilon_b} + \frac{J_c}{\varepsilon_c} \tag{4}$$

   where $J_{a,b,c}$ are the three normal mode actions of the particle as calculated from Wolski Eq. (51) and $\varepsilon_{a,b,c}$ are the three emittances as calculated from Wolski Eq. (30).

   Particles with the largest 6D amplitudes are removed from the bunch to give a "core" bunch.

3. The beam size sigma matrix for the core bunch is analyzed to extract the normal mode eigenvectors and emittances exactly as in the first step.

4. Starting with whole bunch, but using the core bunch eigenvectors and emittances, particles that have the largest 6D amplitudes are removed from the bunch to give a new core bunch. The reason for going through the winnowing process twice is to minimize, to a good degree, any influence that the non-core particles will have on the calculation.

5. The sigma matrix of this new core bunch is computed and, from the sigma matrix, "unnormalized" core emittances $\varepsilon_{\text{core}}(\text{unnorm})$ are calculated.

6. The core emittance values, as calculated in the last step, will be smaller than the whole beam emittances since large amplitude particles are ignored. One way to think about emittance is that it is a measure of the area of phase space a beam inhabits so smaller core emittance values are to be expected. However, such core emittance values are not very insightful. To get around this, core emittance values are normalized so that *if* the beam had a Gaussian distribution, the normalized core emittance values will be the same as the emittance values of the whole beam. One way to think about this is to view the normalized core emittance values as a measure of the density of particles at the core of the beam. Since the core density is independent of the distribution in the tails, the normalized core emittance values will be independent of how much of the tails are cutoff. At least this will be true for a Gaussian beam. For an actual beam, the difference between the normalized core emittances and the whole beam emittances will be an indicator of how non-Gaussian the distribution in the core is.

Normalized core emittances $\varepsilon_{\text{core}}(\text{norm})$ values are calculated from the unnormalized values by multiplying by a factor f_n such that, if the beam shape were Gaussian, the value of the normalized core emittance will be the same as the whole beam emittance $\varepsilon_{\text{core}}(\text{norm}) = \varepsilon$. f_n is calculated by first calculating the 6D amplitude cutoff A_C so that, with a Gaussian distribution, the probability of a particle having a 6D amplitude less than A_C is equal to *ltt%core_emit_cuttoff* which here will

be denoted by C. A_C is calculated from the equation

$$\begin{aligned} C &= \iiint_{A \le A_C} dJ_a\, dJ_b\, dJ_c\, \rho(J_a, J_b, J_c) \\ &= \int_0^{A_C} d\widetilde{J}_a \int_0^{A_C - \widetilde{J}_a} d\widetilde{J}_b \int_0^{A_C - \widetilde{J}_a - \widetilde{J}_b} d\widetilde{J}_c\, \exp\left[-\widetilde{J}_a - \widetilde{J}_b - \widetilde{J}_c\right] \\ &= 1 - \left(1 + A_C + \frac{1}{2}A_C^2\right) \exp\left[-A_C\right] \end{aligned} \tag{5}$$

where ρ is the particle probability density and $\widetilde{J}_a = J_a/\varepsilon_a$, etc. are normalized actions. Given some value for C, A_C can be calculated from Eq. 5 using a Newton search. For a Gaussian distribution, the unnormalized core emittance for, say, the a-mode will be

$$\varepsilon_{a,\text{core}}(\text{unnorm}) = \frac{1}{C} \iiint_{A \le A_C} dJ_a\, dJ_b\, dJ_c\, J_a\, \rho(J_a, J_b, J_c) \tag{6}$$

Doing this integral and combining everything gives the normalization factor which is the same for all modes

$$f_n = \frac{\varepsilon_{\text{core}}(\text{norm})}{\varepsilon_{\text{core}}(\text{unnorm})} = \frac{\varepsilon}{\varepsilon_{\text{core}}(\text{unnorm})} = \frac{C}{1 - \left(1 + A_C + \frac{1}{2}A_C^2 + \frac{1}{6}A_C^3\right) \exp\left[-A_c\right]} \tag{7}$$

14.2 Non-Combined Core Emit Calculation

If *ltt%core_emit_combined_calc* is set to False, the core emittance calculation is modified. With the combined calculation, the set of core particles for a given cutoff value C is the same for all modes. For the non-combined calculation the set of core particles for each mode is calculated independent of the other modes. That is, on a given turn, there are actually three separate calculations, one for each normal mode of oscillation. The analysis is disjoint in that the calculation of the emittance of one mode does not affect the analysis for the other two modes.

The calculation of the core emittance for a given normal mode is:

1. The beam size sigma matrix Σ_{ij} for the whole bunch is analyzed to extract the normal mode eigenvectors as as shown in a paper by Wolski[3].
2. Particles that have the largest amplitudes for the mode under consideration are removed from the bunch to give a core bunch.
3. The beam size sigma matrix for the core bunch is analyzed to extract the normal mode eigenvectors exactly as in the first step.
4. Starting with whole bunch, but using the core bunch eigenvectors, particles that have the largest amplitudes for the mode under consideration are removed from the bunch to give a new core bunch.
5. The sigma matrix of this new core bunch is computed and, from the sigma matrix, a (unnormalized) core emittance for the mode under consideration is calculated.

6. The emittance value as calculated in the last step will be smaller than the whole beam emittance since large amplitude particles are ignored. To compensate for this, the emittance is normalized by a factor f_n as discussed below.

The factor f_n in step 6 above is calculated by first calculating the action cutoff J_c so that, with a Gaussian distribution, the probability of a particle having an amplitude less than J_c is *ltt%core_emit_cuttoff* which here will be denoted by C. J_c is calculated from the equation

$$C = \int_0^{J_c} dJ\, \rho(J) = \int_0^{J_c} dJ\, \frac{1}{\varepsilon}\, \exp(-J/\varepsilon) = 1 - \exp(-J_c/\varepsilon) \tag{8}$$

where ρ is the particle probability density. Solving for J_c gives

$$J_c = -\varepsilon\, \log(1 - C) \tag{9}$$

Assuming a Gaussian distribution, The computed unnormalized core emittance will be

$$\varepsilon_{\text{core}}(\text{unnorm}) = \frac{1}{C} \int_0^{J_c} dJ\, \rho(J)\, J \tag{10}$$

Combining everything gives the normalization factor

$$f_n = \frac{\varepsilon_{\text{core}}(\text{norm})}{\varepsilon_{\text{core}}(\text{unnorm})} = \frac{\varepsilon}{\varepsilon_{\text{core}}(\text{unnorm})} = \frac{C}{1 - (1 + \tilde{J}_c)\, \exp(-\tilde{J}_c)} \tag{11}$$

where $\tilde{J}_c \equiv J_c/\varepsilon = -\log(1 - C)$. The factor f_n will be the same for all three core emittance calculations.

The difference between the combined and non-combined calculations is that with the non-combined calculation, the distribution for any one mode includes particles with large amplitudes for the other two modes. This is not true for the combined calculation.

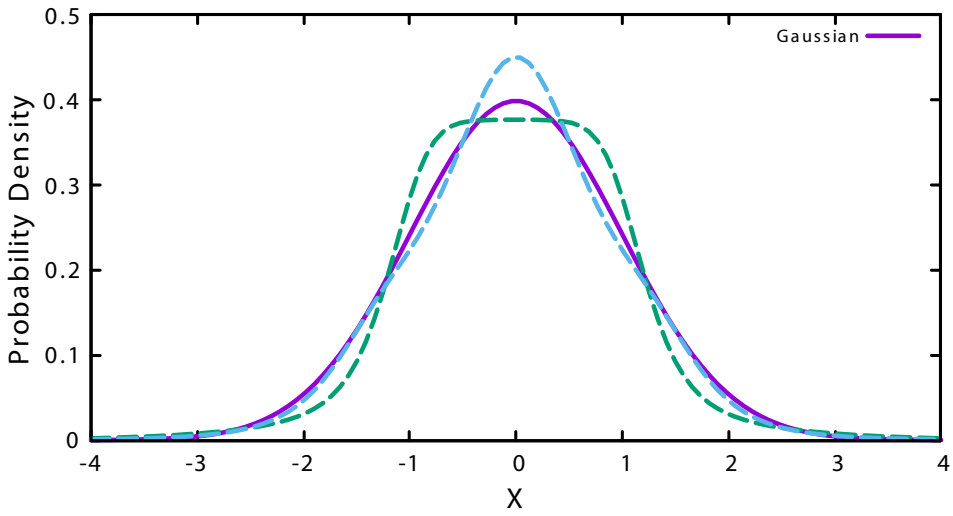

Figure 1

15 kurtosis

The kurtosis κ_{4i} is defined by

$$\kappa_{4i} \equiv \frac{\left\langle dr_i^4 \right\rangle}{\sigma_i^4} - 3, \quad i = x, y, \text{or}, z \tag{12}$$

With this definition,[4] a beam with a Gaussian distribution will have zero kurtosis. A positive kurtosis indicates more particle in the tail than a standard Gaussian and a negative kurtosis indicates less particles.

It is important to keep in mind that the value of the kurtosis says nothing about the distribution of particles near the center of the bunch. This is illustrated in Fig. 1. This figure shows three particle density curves which have been constructed to illustrate this point. The curve drawn with a solid line is a Gaussian and the other two dashed curves are non-Gaussian. Each curve has been constructed to have the same sigma equal to one. The Gaussian curve has a kurtosis of 0 as expected and the other two have a kurtosis equal to 2 due to each having a tail that varies as $1/x^6$ for $|x|$ large. Despite the two non-Gaussian curves having the same kurtosis, the shape of the distributions in the core region are quite different with one being peaked in the center relative to the Gaussian and the other being flattened relative to the Gaussian. To characterize the core, there is the core emittance (§14).

[4] Be aware when reading the literature that there are two definitions of kurtosis in general use. The difference being whether three is subtracted off or not.

16 Particle Data Output

The following describes the particle data output format. Particle data is outputted when the *ltt%simulation_mode* is set to *"BEAM"*.

Particle data is always recorded when the beam is at the *ltt%ele_start* position independent of where the lattice begins and ends.

An output data file will be produced every *ltt%particle_output_every_n_turns* turns. Each line in this file records a particle's orbital and spin position. The name of the data file is derived from the *ltt%phase_space_output_file* or *ltt%action_angle_output_file* string. If the file name contains a hash character "#", the data file name is formed by substituting the turn number for the hash token. If there is no hash character, the data file name is formed by appending the turn number to the file name If there are multiple bunches, instead of using the turn number, the substituted string will be of the form

```
{bunch_index}-{turn_number}
```

Nominally particle data is recorded every *ltt%particle_output_every_n_turns* number of turns. However, if *ltt%particle_output_every_n_turns* is set to 0, particle data is recorded only at the start and end of the tracking. If *ltt%particle_output_every_n_turns* is set to -1, particle data is only recorded at the end of tracking.

17 Spin Tracking Damping Time and Polarization Analysis

Long term tracking including spin is often used to calculate damping times and equilibrium polarizations. One common strategy is to first track a beam for a few damping times until it has reached equilibrium in the orbital phase space. The beam distribution is now saved and a new run is started with this saved orbital distribution and with all the spins aligned so that the beam is 100% polarized. A plot of polarization vs time now gives the damping time. See the section in the Bmad manual on "Polarization Limits and Polarization/Depolarization Rates".

18 Some Checks to be Done if Particles Cannot be Stability Tracked

There can be any number of reasons why particles cannot be stably tracked over many turns. The following are some checks that can be done.

- Check for a resonance condition $l\,Q_x + m\,Q_y + n\,Q_z = p$. The *show universe* command in Tao will show the tunes. The *tune_plane_res_plot* program can be used to plot resonance lines in the (Q_x, Q_y) tune plane. Also the *tune_scan* program can be used to see where the beam blows up in the (Q_x, Q_y) tune plane.
- Check that radiation damping and excitation are both on or both off.

- Check that the RF voltage is sufficient considering the average radiation loss plus the energy sigma of the beam.
- If using PTC or MAP tracking, check that enough slices are being used to accurately model the lattice
- Check that lattice is stable. To do this, check that the damping partition numbers are all positive. The damping partition numbers can be seen using the Tao program's *show universe* command. Note: Make sure that in Tao that the radiation damping and RF on/off match what is being used in the long_term_tracking program.
- Using SINGLE mode tracking, output a turn-by-turn orbit by setting:

```
ltt%phase_space_output_file = "some-nonblank-name"
ltt%particle_output_every_n_turns = 1
```

  Use your favorite plotting program to plot the phase space trajectory. This will give a lot of information. For example, look to see in which plane an instability starts.

References

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

[2] É. Forest, Y. Nogiwa, and F. Schmidt. The FPP and PTC libraries. In Int. Conf. Accel. Phys pp 17–21, (2006).

[3] A. Wolski, "Alternate approach to general coupled linear optics", Phys. Rev. Special Topics, Accel & Beams, **9, 024001 (2006).**

[4] Colwyn Gulliford, Adam Bartnik, Ivan Bazarov, Bruce Dunham, and Luca Cultrera, "Demonstration of Cathode Emittance Dominated High Bunch Charge Beams in a DC gun-based Photoinjector", Appl. Phys. Lett. 106, 094101 (2015). https://doi.org/10.1063/1.4913678