Modeling the g-2 Ring with *BMAD*

D. Rubin
(Dated: February 28, 2022)

CONTENTS

I. INTRODUCTION

The program *g2_tracking* propagates charged particles into and around the g-2 ring. The guide field, including M5 line, injection channel and storage ring, is modelled using element types defined in the *BMAD* programming language. *g2_tracking* is assembled largely from routines in the *BMAD* (fortran) subroutine library.

II. INPUT

There are three fundamental inputs to the program:

1. Model of the guide field. The model is described in the *BMAD* style as 'lattice' files. The lattice file consists of standard 'elements' (dipole magnets, quadrupole magnets, drifts, etc.), 'custom' elements (backleg fringe, inflector, pulsed kicker), and 'markers' with special functions like colimators, detector planes, etc. The elements, characterized by a length, a curvature and a function, are arranged in a 'line' that fixes the geometry. The 'lattice' file is parsed by the *BMAD* subroutine 'bmad_parser' and the data is loaded into fortran structures for easy access. An example of a lattice file that is appropriate for tracking particles from the end of the M5 line, through backleg and inflector and into the ring is `bmad_esquad_grid.`

2. Run specific inputs, (lattice file to read, number of particles to track, number of turns around the ring, initial distribution, initial offsets, peak kicker field, kicker pulse shape, quad field index, etc.) are set in a namelist file 'input.dat', for example, `input.dat`

3. Reference data files. Reference files include magnetic and electric field maps, kicker pulse shape, initial muon distributions, temporal distributions of injected muon pulses, alternative representations of lattice elements, etc. Linux style 'soft links' provide access to reference data files. The current complete set of soft links effective on the Cornell grid is `softlink_lnx`

III. OUTPUT

On execution, the program *g2_tracking* creates a new subdirectory. The name of the subdirectory is the date and time (yearmmdd_hhmmss). All of the output is written to this subdirectory. For reference the input parameter file `input.dat` and the lattice file (i.e. `bmad_esquad_grid.`), are copied into the subdirectory. The output depends of course on the choice of input parameters. All output is written as text files. Available output includes (but is not limited to):

1. Phase space. The complete set of phase space coordinates for each muon in the distribution at a particular location is written to a file named as ⟨LOCATION⟩`_phase_space.dat`. For example, the file `INJ_TO_RING_phase_space.dat` is a list of the phase space coordinates of the particles transported through the injection channel to the end of the inflector, just before transitioning into the ring.

2. Beam moments of the distribution.

   (a) `Beam_moments_tbt.dat` lists the first and second moments (averages and widths) of the phase space of the distribution at the instant of time when the reference particle (magic momentum particle with zero betatron amplitude) arrives at the 'end' of the ring on each revolution. (The 'end' is the last element in the 'lattice line'). Note that all particles in the distribution do not arrive at the end of the ring at the same time. A real detector measures the position of particles in a detector (location) at a particular time. The moments are computed for the entire distribution independent of arrival time. The average arrival time is also included.

   (b) `Beam_moments_by_element.dat` lists the moments at the end of each element in the lattice file.

3. Time binned moments. `Time_Dep_Moments_at_END.dat` lists the moments of the distribution of particles that arrive at the 'end' of the ring within a time bin. The width of the time bin *time_bin*, is specified in the `input.dat` file. Typically *time_bin* is set to 10 ns. Then the moments correspond to what a real (ideal) detector would measure in each 10 ns time bin.

4. Single particle trajectories. If the number of muons (*nmuons* in the input file) is set to 1, then the phase space coordinates of the single particle trajectory are recorded, at the end of each turn in `single_particle_tbt.dat` and at the end of each element in the lattice file `single_particle_by_element.dat`

IV. TRACKING

Particles are propagated through the guide fields defined in the 'lattice' file, by integration of the equations of motion. The electro-magnetic fields are represented by field maps or multipole expansions. The magnetic fields in the injection channel (backleg fringe field and inflector) are described by field maps generated with finite element codes. There are a number of possible representations of the electric fields in the ring quadrupoles, including a field map (generated with OPERA 3D), a multipole expansion in cartesian coordinates (as in the original quad paper) or in cylindrical coordinates based on McMillan functions. The representation of your choice is selected in the 'lattice' file. The multipole representation of the quads assumes that the four plates are at the same voltage. Some analyses rely on independently powering the quad plates, in order, for example, to incorporate quad scraping, and/or voltage errors that are not common to the four plates simultaneously. In such cases it is convenient to use a superposition of the field maps for each of the four plates individually powered. This option is also available by chosing the appropriate lattice file.

The kicker magnetic field can be represented as a field map or as a uniform vertical field. For the temporal profile it is standard practice to use the magnetometer measurement of the time dependence of the kicker field. Alternatively one can specify a perfect rectangular pulse as well as other slightly more realistic versions.

The main dipole field is nominally uniform with field errors included as multipoles.

The M5 line consists of nominally linear magnets (dipoles and quadrupoles). The tracking algorithm takes advantage of that linearity to propagate the phase space through those elements using matrix multiplication.

V. APERTURES

Apertures are defined for each element in the lattice file. With a couple of important expections, particles with trajectories that intersect with the aperture are lost. Consider for example the inflector. The inflector aperture is rectangular with width and height of 18mm and 56 mm resepectively. If a muon trajectory hits that aperture it is lost and dropped from the distribution. The aperture of the quadrupoles is fixed by the quad plates. The collimator aperture is circular with radius 45mm.

Stored particles are confined within the quad and collimator aperture. But particles that exit the inflector en route to the ring must typically pass through the outer plate of Q1. In order to allow for this possibility the Q1 aperture is removed for injected particles, and multiple scattering is included for particles that pass through the quad plate.

In addition, multiple scattering can be 'turned on' for passage through the coils that overlap the beginning and end of the inflector.

VI. INITIALIZATION OF THE DISTRIBUTION

A. Origin

The initial distribution of particles can be read from a file or generated at random according to an algorithm. There are a number of options available.

B. Pre existing distribution

A distribution generated by Dyktys Stratakis (see GM2-doc-4461-v1 from the production target to the end of the M5 line is available with the following instruction in the input file:

- $\langle$muon_file $='$ particles_M4M5End_400_mod.txt$'\rangle$.

The Stratakis distribution is generated by supposing that 8GeV protons are incident on the conversion target all at the same time. While all of the protons strike the target at the same time, the muons that survive transit around the delivery ring and through the M4 and M5 lines emerge with a momentum dependent delay. That delay is on the order of a few nanoseconds. In fact, the length of the proton pulse on target is more than 120ns. That temporal distribution is imprinted on the phase space distribution in an additional step. The parameters of the pulse shape are defined in the input file, most commonly by reference to a measurement. With the setting

- $\langle$tdistr $=$ "e989 $-$ P1"$\rangle$

the reference trace is that measured for the first bunch in the eight bunch train.

C. Generating a random distribution

The phase space coordinates (x, p_x) of the distribution are correlated according to

$$x(s) = \sqrt{a\beta}\cos(\phi(s))$$
$$p_x = p\frac{dx(s)}{ds} = p\frac{\sqrt{a}}{\beta}\alpha\cos\phi(s) + \frac{\sqrt{a}}{\beta}\sin\phi(s)$$

where a is the emittance, $\alpha = \frac{1}{2}\frac{d\beta}{ds}$ and $\frac{d\phi}{ds} = \frac{1}{\beta}$. A distribution with the appropriate correlation is created by using a random gaussian distribution of horizonal and then vertical amplitudes (a), and then a random flat distributon from $0 \to 2\pi$ of phase ϕ.

Parameters in the input file that specify initial phase space distribution

nmuons	= 100000	! number of muons
create_new_distribution	= T/F	!Create a new distribution or read an existing file
seed	=0	!Random number seed. If seed =0 then random number set to system clock.
new_file	= 'empty'	!If you choose to create a new distribution it will be saved in new_file.
muon_file	= file_name	! read the phase space distribution off the target from this file.
tdistr	= "e989-P10"	! Time distribution of the muons, choices are "flat", "gaus", "e821", !"e989" or "e989-P1" through "e989-P8" or the average "e989-P10"
tlength	= 0.	! Width in time of distribution : (doesn't apply to tdistr="e821" or "e989...")
tsigma	= 0.	! standard deviation (width) of time distribution if it is gaussian, !(doesn't apply tdistr="e821" or "e989")
pzdistr	= "gaus"	! "gaus" or "flat"
pz	= 0.01	cutoff of longitudinal momentum distribution, !(fractional energy offset with respect to reference)
pzsigma	= 0.013	! width (sigma) when pzdistr="gaus"
epsdistr	= "gaus"	!emittance distribution at production target: "delta", "flat", "gaus"
epsx	= 16.6e-6	!horizontal emittance at inflector assuming no losses in injection channel (m-rad)
epsy	= 16.6e-6	!vertical emittance at inflector assuming no losses in injection channel (m-rad)

D. Focusing

The beam is focused by the quadrupoles in the M5 line to maximize transmission into the storage ring. The inflector is the most restrictive aperture in the injection channel with full width and height of 18mm and 56mm respectively. By comparison the full width and height in the ring are both 90mm. The mismatch in the horizontal plane is most pronounced. To minimize beam size in the inflector we adjust the focusing in the M5 line so that the beta function in the inflector will be small. But a small beta in the inflector results in a large beta in the ring. There is not much gained if all of the muons clear the inflector only to be lost in the ring. Losses in the inflector and the ring are balanced if at the midpoint of the inflector $\beta_h \sim 2$m and $\beta_v \sim 10$m. In addition to the focusing in the M5 line quadrupoles, the beam is further horizontally defocused on passage through the fringe field of the dipole in the injection channel. The distribution can be initialized with $\beta-$functions and emittance at the entrance to the start of the injection channel so that it will have the desired $\beta_h = 2$m, $\beta_v = 10$m at the midpoint of the inflector.

The parameters in the input file appear as follows:

twiss $= \beta_x, \beta_y, \alpha_x, \alpha_y, \eta_x, \eta_y, \eta'_x, \eta'_y, \phi_x, \phi_y, \gamma_x\gamma_y$
twiss = 2.0, 10.0, 0., 0.0, 0.0, 0., 0., 0., 0.7, 0.7, 0., 0.

twiss_ref ='end' ! calculate beta functions of distrubution so that it will take on specified 'twiss' values at inflector end
twiss_ref = 'center' ! calculate beta functions of distrubution so that it will take on specified 'twiss' values midway through inflector
twiss_ref = 'use_as_is' ! Use specified 'twiss' values directly for the distribution

E. Offsets

The offset and angle of the centroid of the distribution are likewise specified in the input file. If the starting point of the tracking excercise is at the input to the injecton channel (end of the M5 line) then the trajectory for a magic momentum muon with the following offsets at the start of the injection channel will exit the inflector at its horizontal and vertical midpoint with a horizontal angle of -2 mrad.

initial_offsets_ref = -2.9343E-02 0. 0. 1.6292E-02 0. 0. ! $x_{off}, y_{off}, z_{off}, x'_{off}, y'_{off}, z'_{off}$
initial_offsets = -2.9343E-02 0. 0. 1.6292E-02 0. 0.

There is a procedure to find offsets at the start of the injection channel that correspond to offsets specified at the inflector exit. For example, suppose we want the offsets such that the trajectory of the magic momentum muon exit the inflector dead center with radial angle -2mrad.

opt_incident = T T/F
inflector_end_target = 0. 0. 0. -0.002 0. 0. $x, y, z, x', y', \delta p/p$

As written the program does not update 'initial_offsets'. That step is done by hand.

F. Starting point

The longitudinal coordinate of the start of tracking is either at the beginning of the 'line' as defined in the lattice file, or at the inflector exit according to

start_tracking_at_inflector_exit = F ! F - tracking will start at the beginning of the lattice 'line', T- at the inflector exit

The offsets and twiss parameters must be chosen consistent with the starting point. While the initial horizontal offset and angle indicated above (-29.3 mm and 16.3 mrad) are a reasonable choice if tracking begins at the input to the injection channel, -77 mm and -2mrad would be the corresponding choice for tracking that begins at the inflector exit. Likewise the initial twiss parameters depend on the starting point.

VII. EXAMPLES

A. Create Distributions at Inflector Exit

Propagate a distribution from the end of the M5 line and through the injection channel to the inflector exit. The centroid of the distribution corresponds to the trajectory of the magic momentum muon. Choose the initial offsets so that the magic momentum muon exits the inflector with $x_{inf} = 0, x'_{inf} = -1\text{mrad}, y_{inf} = y'_{inf} = 0$.

1. The first step is to find the initial phase space coordinates of the magic momentum muon. The directory /home/dlr/development9_linux/g-2/examples/example11 contains the necessary files including
   (a) /home/dlr/development9_linux/g-2/examples/example11/bmad_esquad_grid. - lattice file.
   (b) /home/dlr/development9_linux/g-2/examples/example11/input.dat - run specific parameters. Note that

   ```
   inflector_end_target = 0. 0. 0. -0.001 0. 0. !x,y,z,xp,yp,dp/p
   opt_incident =         T                     !find initial_offsets that yield inflector_end_target
   ```

   (c) All of the soft links to the reference files are set with the command
   `/home/dlr/development9_linux/g-2/softlink_lnx`
   (d) Not to be forgotten, quad parameters are set in 'quad_input.dat' and 'quad_plate_misalign.bmad'
   (e) To execute the latest version of the program type `/home/dlr/development9_linux/production/bin/g2_tracking`
   (f) The search for the vector of `initial_offsets` that leads to the desired `inflector_end_target` vector begins with the `initial_offsets` that are read from the 'input.dat' file. If the search fails the choice of `initial_offsets` may need to be adjusted.
   (g) The output is written to `subdir/optimize_incident_out.dat` (`subdir` is the subdirectory generated for the run and named with date and time `year/mm/dd_hhmmss`
   Here is an example

```
[dlr10@lnx6186 example11]$ more 20220225_144619/optimize_incident_out.dat
  element at end of injection line = MARK_INFLECTOR_DS
 iteration       x_init     xp_init     x_final    xp_final
         1 -2.9343E-02  1.6292E-02  4.2847E-07 -1.9998E-03
 end_orb%vec(1:2) =   3.8743E-10 -1.0000E-03
 iteration       x_init     xp_init     x_final    xp_final     Delta x    Delta xp
         2  -3.7440E-02  1.8816E-02 -3.3623E-04 -1.1726E-03 -8.0968E-03  2.5240E-03
         3  -3.7549E-02  1.8892E-02 -2.3286E-07 -9.9990E-04 -1.0870E-04  7.5782E-05
         4  -3.7547E-02  1.8891E-02 -1.6138E-08 -1.0000E-03  1.6958E-06 -5.1047E-07
         5  -3.7547E-02  1.8891E-02 -2.1031E-08 -1.0000E-03  7.8237E-09 -4.1334E-10
         6  -3.7547E-02  1.8891E-02  9.9876E-09 -9.9999E-04 -5.2470E-08  1.9456E-08
         7  -3.7547E-02  1.8891E-02  2.8900E-09 -1.0000E-03  4.4945E-08 -1.5309E-08
         7  -3.7547E-02  1.8891E-02  2.8900E-09 -1.0000E-03  4.4945E-08 -1.5309E-08

initial_offsets = -3.7547E-02 0.0 0.0  1.8891E-02 0.0 0.0
```

```
initial_offsets_ref = -3.7547E-02 0.0 0.0  1.8891E-02 0.0 0.0
```

(h) Replace the 'initial_offsets' and 'initial_offsets_ref' in the 'input.dat' file with the last two lines of 'optimize_incident_out.dat' Then rerun `g2_tracking`. The output to the monitor will include the following:

```
------------------------------------------------------------------------------
Fractional Tune:        Qx = 0.94917   Qy = 0.31776
RING PARAMS:        Qx = 0.94918   Qy = 0.31776   betax = 7.64553   betay =22.02755   field_index
ELE(0)%beta:        betax = 7.64553   betay =22.02755
 circumference =   4.46860139E+01 radius =   7.11200000E+00
start_orb -3.7547E-02  1.8891E-02  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
start_orb ref -3.7547E-02  1.8891E-02  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  element at end of injection line = MARK_INFLECTOR_DS
 Optimize_Incident: err_flag =   F  track_state =          -9
 iteration       x_init     xp_init     x_final    xp_final
         1 -3.7547E-02  1.8891E-02 -2.7141E-06 -1.0013E-03
 end_orb%vec(1:2) =   3.8743E-10 -1.0000E-03
         2  -3.7547E-02  1.8891E-02 -1.4181E-08 -1.0000E-03  1.4369E-07  2.8932E-07
         3  -3.7547E-02  1.8891E-02  3.6762E-08 -9.9998E-04 -1.4435E-08  6.3270E-09
         4  -3.7547E-02  1.8891E-02 -3.7868E-08 -1.0000E-03  1.7425E-08 -9.9100E-09
         5  -3.7547E-02  1.8891E-02  4.3686E-08 -9.9997E-04 -2.3444E-08  1.2164E-08
         6  -3.7547E-02  1.8891E-02 -3.2143E-12 -1.0000E-03  3.6329E-08 -1.6623E-08
         6  -3.7547E-02  1.8891E-02 -3.2143E-12 -1.0000E-03  3.6329E-08 -1.6623E-08
  trajectory through backleg and inflector
  0.0000E+00 -3.7547E-02  1.8891E-02  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00 BACKLEG_START
  2.4801E+00  8.0807E-03 -5.6106E-03  0.0000E+00  0.0000E+00 -4.5804E-04  0.0000E+00 FRINGE
  2.4801E+00  8.0807E-03 -5.6106E-03  0.0000E+00  0.0000E+00 -4.5804E-04  0.0000E+00 MARK_CRYO_US
  2.6000E+00  6.7930E-03 -1.5562E-02  0.0000E+00  0.0000E+00 -4.6550E-04  0.0000E+00 CRYO_US
  2.6000E+00  6.7930E-03 -1.5562E-02  0.0000E+00  0.0000E+00 -4.6550E-04  0.0000E+00 MARK_INFLECTO
  4.3000E+00 -2.7141E-06 -1.0013E-03  0.0000E+00  0.0000E+00 -4.8822E-04  0.0000E+00 INFLECTOR
  4.3000E+00 -2.7141E-06 -1.0013E-03  0.0000E+00  0.0000E+00 -4.8822E-04  0.0000E+00 MARK_INFLECTO
  4.3000E+00  7.6997E-02 -1.0013E-03  0.0000E+00  0.0000E+00 -4.8822E-04  0.0000E+00 INJ_TO_RING
  4.3000E+00  7.6997E-02 -1.0013E-03  0.0000E+00  0.0000E+00 -4.8822E-04  0.0000E+00 RING_BR
  4.3000E+00  7.6997E-02 -1.0013E-03  0.0000E+00  0.0000E+00 -4.8822E-04  0.0000E+00 END
```

The very last line $s, x, x', y, y', z, dp/p$ is the phase space of the trajectory at the end of the inflector. The trajectory exits the inflector at the midpoint of its aperture (77mm from the magic radius), with an angle of -1mrad.

2. The next step is to track each particle in a distribution through the injection channel. Probably the most straightforward way to proceed is to read an existing distribution. Diktys, Volodya, and Eremery have generated such distributions, some of which are accessed by the preset soft links. But as those files have a limited number of particles, and as the efficiency of capturing such a particle in the ring is on the order of 1%, that number is often insufficient. Alternatively we can generate our own particle list assuming gaussian distributions as discussed earlier. The files in directory /home/dlr/development9_linux/g-2/examples/example12 are set up to:

   (a) create a distribution of 100,000 muons at the start of the injection channel

   (b) propagate the particles in that distribution into the ring

   (c) write the phase space coordinates of the particles that arrive at the inflector exit to a file

   The program has been executed in this directory (`example12/`), and written output to `subdir` = 20220225_152556. The phase space coordinates of the particles at the end of each of the elements in the 'line' are written to files named ⟨ELEMENT_NAME⟩`_phase_space.dat`. The file `INJ_TO_RING_phase_space.dat` is the distribution at the inflector exit.

   Multiple runs can be combined as necessary.

B. Momentum Capture

The distribution `INJ_TO_RING_phase_space.dat` is propagated into the ring where each particle receives a kick from the pulsed kicker. As the kicker is not uniform, the received kick depends on arrival time of the particle. The momentum bite that is captured depends on the kick. The result is that the captured momentum bite depends on the arrival time. We define capture as survival for at least 4 microseconds (about 28 revolutions). We delete all of the particles that are lost in those 4 microseconds from the initial distribution. The result is the momentum and time of the particles that survive. The files in /home/dlr/development9_linux/g-2/examples/example13 are configured to that end. Note that in the `input.dat` file

```
nturns  = 100
nmuons = 43000
create_new_distribution = F
muon_file = 'INJ_TO_RING_phase_space.dat'   !The intitial distribution at the inflector exit
                                                ! is from our calculation in {\tt example12/}
twiss_ref = 'use_as_is'   !Use the distribution as is, without modification of beta functions
start_tracking_at_inflector_exit = T        !Start tracking at the inflector exit
initial_offsets_ref =  0. 0. 0. 0. 0. 0.    !Use the distribution as it is at the inflector exit
initial_offsets =  0. 0. 0. 0. 0. 0.
kickerPulseFile = 'v0/traces-Ir.txt'            !kicker pulse trace
kicker_params%kicker_field      =   204.2e-4   204.2e-4  204.2e-4 ! Peak field on each of the 3 kickers
kicker_tStart            =   120.0e-9    120.0e-9   120.0e-9   ! kicker relative time
scraping_on = T             !quad scraping is turned on
write_phase_space_every_n_turns = 28  !The phase space of the distribution after 28 turns
```

The files generated by that last line are, among others,
/home/dlr/development9_linux/g-2/examples/example13/20220226_204947/END028_phase_space.dat and
/home/dlr/development9_linux/g-2/examples/example13/20220226_204947/END000_phase_space.dat

C. Combining g2_tracking output

Another program `/home/dlr/development9_linux/production/bin/energy_vs_time` reads the file with the distribution that has survived 28 (END028_ . . .) turns and the starting distribution (END00_...) to generate the captured time-momentum distribution as follows:
In directory /example13 type on the command line

```
{\tt /home/dlr/development9\_linux/production/bin/energy\_vs\_time}
```

The program will look for the phase space files named `END028_phase_space.dat` and `END000_phase_space.dat` in all of the date named subdirectories (here there is only one), combine them and write `/example13/all_Energy_vs_time_0.dat`

VIII. BATCH SUBMISSION

It is often convenient to take advantage of the CLASSE compute farm. If the input file is configured to create a new distribution, and `seed=0`, then we simply submit the job on multiple computers and then combine the results. An example batch submission is shown in `/nfs/gm2/data2/dlr10/g-2/mytest/energy_vs_time_vs_turn/test/` Note that the input file `test/input.dat` is configured to generate a new distribution (as in `/example12` and that tracking starts at the start of the injection channel. See the Classe computer group grid engine wiki page for more about batch submission, handy commands, etc.

With the command
`./submit_g2_10.sh  g2.sh`
ten jobs are submitted to the grid.

Finally run `energy_vs_time` to read the data from all of the date named subdirectories and write `all_Energy_vs_time_0.dat`.