Summary of work completed on a muon tracking program for the development of the g-2 kicker

Robin Bjorkquist

August 2011

Contents

1 Overview of program

I have been working on writing a program to simulate the passage of muons through the storage ring of the new g-2 experiment, with the goal of improving the design of the kicker magnet (see the project proposal [1] for general information about the experiment; Ref. [2] has a wealth of information about the previous version of the experiment at Brookhaven).

The programming is done in Fortran 90. At the beginning of the summer, I learned the language from a book by Metcalf and Reid [3], and have often referred back to it during my time working on this project; I warmly recommend it to anyone with questions about Fortran. I also learned a great deal from *Numerical Recipes in Fortran* [4], both about programming in Fortran and about methods of scientific computation; my program uses *Numerical Recipes* routines to solve ordinary differential equations and generate gaussian distributions.

The main program is called `muon_trajectory`, currently in the 11th version. There is also a `parameters` module where all constants and physical dimensions are defined, and a `derivs` subroutine that generates the azimuthal derivatives of the muon's position and momentum coordinates, for use by the *Numerical Recipes* `odeint` subroutine. Here is a list of things the main program does:

- Reads in the various input parameters.
- Converts those inputs (as needed) to the dimensionless variables used throughout the internal calculations of the program.
- Generates gaussian distributions for the initial horizontal and vertical position, momentum, and start time of the muons.
- Loops over the total number of muons, tracking each muon in turn through the storage ring.
- Calculates the angular position at which the muon will strike the outer plate of the first quadrupole.
- Tracks the muon from the inflector end to the point where it strikes the plate using the *Numerical Recipes* `odeint` routine.
- Applies a (gaussian-distributed) scattering angle to the muon's trajectory, to account for its passage through the aluminum quadrupole plate.
- Tracks the muon around the ring the specified number of times using the *Numerical Recipes* `odeint` routine.
- Writes trajectory information for the muon to an output file.

Now, more details for each of these items:

2 Differential equations

The heart of the muon-tracking program is a set of differential equations. These equations specify how the position and momentum of a charged particle change under the influence of a given electromagnetic field. `Muon_trajectory_11` uses the `odeint` subroutine from the LEPP library of *Numerical Recipes* routines to solve these equations numerically, thereby generating a trajectory for a muon passing through the g-2 storage ring.

The *Numerical Recipes* `odeint` routine solves a specified set of ordinary differential equations using a fifth-order Cash-Carp Runge-Kutta algorithm and adaptive stepping (see Ref. [4] Section 16.2 for a discussion of the algorithm). The differential equations for a charged particle traveling through an electromagnetic field are contained in the `derivs` subroutine.

Here is a derivation of the differential equations. It is a set of six equations, for the three position components and the three momentum components.

The position equations are what you would expect (keeping in mind that the muons travel at relativistic speeds, so that $\mathbf{p} = \gamma m \mathbf{v}$, not simply $m\mathbf{v}$):

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{v}(\mathbf{t}) = \frac{\mathbf{p}(\mathbf{t})}{m\gamma(t)}, \tag{1}$$

and the momentum equations come from the Lorentz Force Law:

$$\frac{d\mathbf{p}(t)}{dt} = q\mathbf{E}(\mathbf{x},t) + q\mathbf{v}(t) \times \mathbf{B}(\mathbf{x},t) \tag{2}$$

$$= q\mathbf{E}(\mathbf{x},t) + q\frac{\mathbf{p}(t)}{m\gamma(t)} \times \mathbf{B}(\mathbf{x},t) \tag{3}$$

We can write γ in terms of the momentum,

$$\frac{1}{\gamma^2} = 1 - \left(\frac{v}{c}\right)^2 = 1 - \left(\frac{p}{mc}\right)^2 \frac{1}{\gamma^2} \tag{4}$$

$$\frac{1}{\gamma^2} = \frac{1}{1 + (p/mc)^2}, \tag{5}$$

so that the set of differential equations becomes:

$$\begin{cases} \dfrac{d\mathbf{x}(t)}{dt} = \dfrac{\mathbf{p}(\mathbf{t})}{m\sqrt{1+(p/mc)^2}} \\ \dfrac{d\mathbf{p}(t)}{dt} = q\left[\mathbf{E}(\mathbf{x},t) + \dfrac{\mathbf{p}(t)}{m\sqrt{1+(p/mc)^2}} \times \mathbf{B}(\mathbf{x},t)\right]. \end{cases} \tag{6}$$

These are not quite the differential equations that appear in `muon_trajectory_11`, because:

1. I convert the equations to dimensionless form, so that the various constants (m, q, c) are absorbed into the coordinates.

2. I use curvilinear coordinates instead of cartesian coordinates, which complicates the appearance of the equations (but very much simplifies the actual calculations!).

3. I use the azimuthal position in the storage ring as the independent coordinate instead of time. This is natural because all the muons enter the storage ring from the fixed inflector end, but they do not all enter at the same time. Also, the various components of the ring (e.g., the kicker plates), are placed at definite locations in the ring.

Item 1 suggests itself when you shuffle around the constants and rewrite the differential equations like this:

$$\begin{cases} \dfrac{d\left(\mathbf{x}/cT\right)}{d\left(t/T\right)} = \dfrac{(\mathbf{p}/mc)}{\sqrt{1+(p/mc)^2}} \\ \dfrac{d(\mathbf{p}/mc)}{d(t/T)} = \left(\dfrac{\mathbf{E}}{mc/qT}\right) + \dfrac{(\mathbf{p}/mc)}{\sqrt{1+(p/mc)^2}} \times \left(\dfrac{\mathbf{B}}{m/qT}\right), \end{cases} \tag{7}$$

where T is some time unit (I use $T = 1$ nanosecond). If we are careful to express t in terms of those time units, lengths in units of cT, momenta in units of mc, electric fields in units of mc/qT and magnetic fields in units of m/qT, we can write the differential equations more simply as:

$$\begin{cases} \dfrac{d\mathbf{x}(t)}{dt} = \dfrac{\mathbf{p}(t)}{\sqrt{1+p(t)^2}} \\ \dfrac{d\mathbf{p}(t)}{dt} = \mathbf{E}(\mathbf{x},t) + \dfrac{\mathbf{p}(t) \times \mathbf{B}(\mathbf{x},t)}{\sqrt{1+p(t)^2}}. \end{cases} \tag{8}$$

This is what I do in `muon_trajectory`, so that the physical constants only need to be invoked at the very beginning (to convert the inputs to my units) and at the very end (to convert the outputs to whatever units are desired), and not at each step of the numerical tracking routine.

Next (item 2), I write the differential equations in terms of my particular choice of coordinates. The position and momentum coordinates I use are

- x, the radial position measured from the center of the beam pipe (so that if the muon is exactly 7.112 m away from the center of the storage ring, $x = 0$).

- y, the vertical position.

- θ, the azimuthal angle around the storage ring, increasing clockwise (the direction the muons move) with $\theta = 0$ at the inflector end. If $R = 7.112$ m is the radius of the storage ring, the distance traveled around the ring is $z = R\theta$.

- p_x, radial momentum.

- p_y, vertical momentum.
- p_z, azimuthal momentum. Note: in `muon_trajectory_11`, I actually store and propagate the *difference* between the azimuthal momentum and the desired "magic momentum," because the Runge-Kutta algorithm involves adding small increments to the values of the coordinates; the computation is more accurate if you are working with a small quantity to begin with, because you get less roundoff error.

I believe these coordinates are standard for accelerators. With regards to the differential equations, it is important to note that x, y and z are curvilinear coordinates, not cartesian. In particular, the radial and azimuthal unit vectors ($\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$) change direction when θ changes. If we express some generic vector v in these coordinates,

$$\mathbf{v} = v_x\hat{\mathbf{x}} + v_y\hat{\mathbf{y}} + v_z\hat{\mathbf{z}}, \tag{9}$$

and take the time derivative, we get

$$\frac{d\mathbf{v}}{dt} = \left(\frac{dv_x}{dt}\hat{\mathbf{x}} + v_x\frac{d\hat{\mathbf{x}}}{d\theta}\frac{d\theta}{dt}\right) + \left(\frac{dv_y}{dt}\hat{\mathbf{y}}\right) + \left(\frac{dv_z}{dt}\hat{\mathbf{z}} + v_z\frac{d\hat{\mathbf{z}}}{d\theta}\frac{d\theta}{dt}\right) \tag{10}$$

$$= \left(\frac{dv_x}{dt}\hat{\mathbf{x}} + v_x\frac{d\theta}{dt}\hat{\mathbf{z}}\right) + \left(\frac{dv_y}{dt}\hat{\mathbf{y}}\right) + \left(\frac{dv_z}{dt}\hat{\mathbf{z}} - v_z\frac{d\theta}{dt}\hat{\mathbf{x}}\right) \tag{11}$$

$$= \left(\frac{dv_x}{dt} - v_z\frac{d\theta}{dt}\right)\hat{\mathbf{x}} + \left(\frac{dv_y}{dt}\right)\hat{\mathbf{y}} + \left(\frac{dv_z}{dt} + v_x\frac{d\theta}{dt}\right)\hat{\mathbf{z}}. \tag{12}$$

With the origin at the center of the storage ring, if $\mathbf{v}$ is the position, we have $v_x = R + x$, $v_y = y$, $v_z = 0$; if $\mathbf{v}$ is the momentum, we have $v_x = p_x$, $v_y = p_y$, $v_z = p_z$, so the differential equations (8) become:

$$\begin{cases} \dfrac{dx}{dt} = \dfrac{p_x}{\sqrt{1+p^2}} \\ \dfrac{dy}{dt} = \dfrac{p_y}{\sqrt{1+p^2}} \\ \dfrac{d\theta}{dt} = \dfrac{p_z}{(R+x)\sqrt{1+p^2}} \\ \dfrac{dp_x}{dt} = E_x + \dfrac{p_yB_z - p_zB_y}{\sqrt{1+p^2}} + \dfrac{p_z^2}{(R+x)\sqrt{1+p^2}} \\ \dfrac{dp_y}{dt} = E_y + \dfrac{p_zB_x - p_xB_z}{\sqrt{1+p^2}} \\ \dfrac{dp_z}{dt} = E_z + \dfrac{p_xB_y - p_yB_x}{\sqrt{1+p^2}} - \dfrac{p_xp_z}{(R+x)\sqrt{1+p^2}} \end{cases} \tag{13}$$

Finally (item 3), I change variables so that θ is the independent variable instead of t. This is the form of

the equations programmed into `derivs.f90`:

$$\begin{cases} \dfrac{dx}{d\theta} = \dfrac{(R+x)p_x}{p_z} \\ \dfrac{dy}{d\theta} = \dfrac{(R+x)p_y}{p_z} \\ \dfrac{dt}{d\theta} = \dfrac{(R+x)\sqrt{1+p^2}}{p_z} \\ \dfrac{dp_x}{d\theta} = \dfrac{(R+x)}{p_z}\left[E_x\sqrt{1+p^2} + p_yB_z - p_zB_y\right] + p_z \\ \dfrac{dp_y}{d\theta} = \dfrac{(R+x)}{p_z}\left[E_y\sqrt{1+p^2} + p_zB_x - p_xB_z\right] \\ \dfrac{dp_z}{d\theta} = \dfrac{(R+x)}{p_z}\left[E_z\sqrt{1+p^2} + p_xB_y - p_yB_x\right] - p_x. \end{cases} \tag{14}$$

3 Muon tracking

With the help of the `odeint` routine, tracking the muons through the storage ring is straightforward, provided that you have some key information. It requires that you provide:

- Values for the initial position, time, and momentum coordinates.
- The electric and magnetic fields inside the storage ring.
- Information about the physical dimensions of the storage ring components, in case the muons hit something.

The initial coordinates are generated according to gaussian distributions, using the *Numerical Recipes* `gasdev` routine (see Ref. [4], Section 7.2).

The electric and magnetic fields in the ring are defined piecemeal, based on whether the muon is currently located between the kicker plates (where there is a magnetic field), between the quadrupole plates (where there is an electric field), or neither (where there is only the ever-present dipole field of the storage ring. I make no attempt to model fringe fields, with the result that the electromagnetic fields defined within the `derivs` subroutine have discontinuities at the boundaries of each storage ring element. I had a great deal of difficulty trying to get the `odeint` routine to cross those discontinuities (it seems that the Runge-Kutta algorithm requires the differential equations to have a certain degree of continuity/differentiability). Eventually, I gave up on doing the tracking in one fell swoop; instead, `muon_trajectory_11` calls `odeint` separately for each element of the storage ring. It makes the code more clunky, but it works!

For more information about the kicker field, see Section 6; for more information about the quadrupole field, see Section 7; the strength of the storage ring dipole field is one of the values stored in the `parameters` module (see Section 5). All these fields are combined together in the `derivs` subroutine, for use in the differential equations (Equation 14).

As for the final item—information about the spaces that are available for muon passage, and the spaces that are *not* available—`muon_trajectory_11` does not include this information. As it stands, the program doesn't know whether a muon is inside the storage ring or outside it; it blindly applies the electromagnetic fields based on the angular position of the muon, with no regard to the location of the beam pipe walls or the quadrupole plates. This is something that will need to be added (see Section 10).

4 Program inputs

I have written an input file (cleverly named "input") that contains the various quantities read in by `muon_trajectory_11`. These quantities are:

- **nturns**, the number of turns through which the program will track the muons; it starts at the opening of the inflector and ends just after the end of the first quadrupole region, so it actually tracks the muons for the specified number of turns plus a little bit more. The reason for the extra bit of tracking is that the first quadrupole region is treated differently during the muons' first turn because the muons have to pass *through* the outer quadrupole plate; after that abnormal section, the program simply tracks the muons around the ring **nturns** times. It would be possible to modify the program so that it stops at the inflector, but I didn't think it was worth the trouble.

- **eps**, the specified accuracy of each Runge-Kutta step taken by the `odeint` routine. See Ref. [4], Section 16.2 for a description of what this actually means. Lately, I have been using eps = 10^{-10}. From comparing trajectories with different values of eps, I get the sense that the value doesn't matter much, as long as it is smaller than about 10^{-4}.

- **h1**, the estimated first step size (in radians) for the `odeint` routine. I picked a value of 0.1 arbitrarily—it doesn't matter much because the `odeint` routine calculates the error of each step and reduces or increases the stepsize accordingly; all it needs is some value to start with.

- **hmin**, the minimum stepsize (in radians) for the `odeint` routine. If the demanded accuracy (set by eps) forces the stepsize to become smaller than this, the program will terminate. I have it set to 0.0 (no limit).

- **dxsav**, how often you want the `odeint` routine to save data points. This does not affect the stepsize taken by the routine. At the end of each Runge-Kutta step, the routine will store the data point if it has been longer than **dxsav** (in radians) since the last data point was saved. Note: inside `muon_trajectory_11`, the logical variable **save_steps** controls whether `odeint` will save data at all. I have it set to true. If you set it to false instead, data will only be stored at the end of each element of the ring (that is, the end of each call to `odeint`). If it becomes desirable to have the capability to turn **save_steps** on and off at will, it would be good to include it in the input file, so that it could be changed without recompiling the program. So far, I have always had **save_steps** set to true.

- **nmuons**, the number of muons to track. I have only run the program for up to 20 muons at a time. Ultimately, we will want to track thousands (if not hundreds of thousands) of muons at once, to get a good sampling of the phase space. I don't know how long the program will take to run under those circumstances, or how much space the data files will take up. Everything works just fine for 20 muons. Note: to track just one muon with a specified set of input coordinates, set **nmuons** = 1, all the standard deviations to zero, and the means to your desired coordinates (I do this when I want to see what happens to the "perfectly placed" muon at the very center of the bunch).

- The mean and standard deviation for each input coordinate (radial position, vertical position, time, and three momentum components). Position is measured in meters, time in nanoseconds, and momentum in units of *mc*. Each of the muons is assigned initial coordinates according to a gaussian distribution. For testing, I have chosen the widths of the distributions somewhat arbitrarily; we will need to work out exactly what the distributions should look like for the actual storage ring. It could well be that the distributions *aren't* gaussian, or are correlated with each other in some way. In that case, the program will require modification. The radial position has a mean of 77 mm, because the end of the inflector is 77 mm farther out than the desired orbit radius of 7.112 m. The other five coordinates all have mean = zero (note that for the azimuthal momentum component, the program actually stores & tracks the *difference* between the azimuthal momentum and the desired "magic momentum"). The x and y deviations will be related to the size of the inflector opening. I have the time deviation set to

25 ns, according to Ref. [5], Section 2.1, but it may be different at Fermilab. I don't know what to expect of the momentum deviations, nor do I know where to look for this information.

- **kickermaxtime**, the time at which the kicker magnet reaches its peak magnetic field. The clock starts when the center of the muon bunch enters the ring. Currently I am using a value of 37 nanoseconds, because that is how long it takes a muon to travel from the end of the inflector ($\theta = 0$) to the center of the kicker plates ($\theta = \pi/2$). The time and magnitude of the kicker pulse are input parameters so that we can adjust them and observe the effect on the muon beam. In later versions of the program, there will probably be other adjustable kicker parameters as well.

- **kickermagnitude**, the magnetic field (in Teslas) produced by the kicker when it is at its maximum. Recently I have been using a value of 0.014 T, because in some earlier version of the program (possibly without the electrostatic quadrupoles programmed in) that turned out to be the optimal value. Ref. [5], page 11 tells me that the required field integral is about 0.1 Tm, and the kicker plates are about 5 m long, so a magnetic field of 0.02 T is in the right ballpark for this value.

- **directory**, the filepath for the directory where the trajectory files will be saved. You have to create this directory before running the program. You will get an error message if the directory already contains trajectory files (it won't overwrite old files).

Immediately after reading in the input file, `muon_trajectory_11` converts the input lengths from meters to the length-unit used by the program, and the input magnetic field from Teslas to the magnetic-field-unit used by the program. Input times are left as nanoseconds, and input momenta are left in units of mc. If you are adding more inputs to the program, be sure that the inputs end up in the correct units for the program calculations. There is a warning about this in the comments at the start of `muon_trajectory_11` and at the start of the `derivs` subroutine. For an explanation of the units used by the program, see section 2 of this report. Appropriate conversion factors are stored in the `parameters` module, as described in section 5.

5 Program parameters

The `parameters` module contains all physical constants, dimensions, and conversion factors used by the `muon_trajectory` program. Values are stored here so that

1. they will be accessible to both the main program and the `derivs` subroutine, without needing to be passed between the two as arguments,

2. any value can be changed in a single location, thereby updating the entire program, and

3. the program user can look in one place to see what values are being used for everything.

Here are some comments on the various values in the parameters module:

physical constants I got the values of these constants from the Particle Data Group [6]. They are used to define the dimensionless units used by the program, as discussed in section 2.

conversion factors I define conversion factors for

- Seconds $\longleftrightarrow$ program time-units (ns). If you decided to use some other time-unit, you could update this value, and any other conversion factors which depend on the time-units you choose would be updated automatically. The notation "timeunit_s" means "this is how many seconds are in one program time-unit." The names of the other conversion factors should be interpreted similarly.

- Meters $\longleftrightarrow$ program length-units (cT, where T = the program timeunit).

- MeV/c $\longleftrightarrow$ program momentum-units (mc).

- Teslas $\longleftrightarrow$ program magnetic-field-units (m/qT).
- Volts $\longleftrightarrow$ program voltage-units (mc^2/q)

These conversions are used to convert parameters and input coordinates to the units used internally by the program, and to convert outputs to whatever units are desired. If more conversion factors are needed, they can be added in the same manner.

storage ring parameters The radius and magnetic field of the storage ring, which is designed for muons traveling at the so-called "magic momentum." (Note: in my units, $p = RB$.)

quadrupole parameters Parameters related to the placement, size, and electric field of the electrostatic quadrupoles. My information about the quadrupoles comes from an article about the Brookhaven g-2 quadrupoles [7]. Here are some comments on the values and what they're used for:

- each quadrupole fills 39 degrees of the ring (Ref. [7], Section 4.1.1); I ignore the structure of the quadrupole (it's actually made in three sections, installed in two separate vacuum chambers), and treat the whole 39 degrees as the same.
- There are four quadrupoles, placed symmetrically around the ring.
- From the angular length and placement of each quadrupole, I calculate the start and end position of each (in radians). This is used by the tracking program when it calls `odeint` for each region of the ring—it has to know where each quadrupole begins and ends.
- The separation of the quadrupole plates is 10 cm. This is used in the calculation of where the muons will hit the quadrupole plate in their first turn around the ring. Presumably, it will also be used when the program includes some kind of aperture check.
- The outer plate of quadrupole #1 is 0.1 mm thick (Ref. [7], Section 4.1.4). This is used to calculate the scattering angle when the muons pass through the plate.
- The scattering angle depends not simply on the thickness of the material, but the thickness divided by the radiation length of that material. The radiation length for aluminum is 8.897 cm (Ref. [6], Section 6).
- The values for the multipole coefficients of the electrostatic field produced by the quadrupoles (Ref. [7], Table 5). My values are the negatives of those given, because the table was generated for the negative muon storage polarity, and this experiment will use positive muons. See section 7 for more discussion of the quadrupole field.

kicker parameters Parameters related to the placement, size, and magnetic field of the kicker. My information about the kicker comes from Ref. [5]. Here are some comments:

- The kicker plates are centered 1/4 of the way around the ring from the inflector. The precise location of the kicker is something that could potentially be adjusted a bit; it also might be possible to add another kicker directly across the ring from the first.
- The kicker plates are 5.28 m long (Ref. [5], Section 2.1), which I use to find the angle the kicker takes up in the ring. I ignore the fact that the kicker is actually made of three sections (each 1.76 m long), and treat it as though the magnetic field is uniform throughout that 5.28-meter-long section of storage ring, with no fringe field.
- Just as for the quadrupoles, I use the kicker center and the storage ring angle it fills to calculate the start and end position in radians.

- The kicker has the time dependence of an RLC circuit. I got values of R, L, and C from Ref. [5] Table 1, and use them to calculate the time dependence. The actual calculation is done in the `derivs` subroutine. Note that the values are converted so that they use nanoseconds instead of seconds.
- **kickermaxtime** and **kickermagnitude** are listed here simply so that the `derivs` subroutine and the main program will have access to the values. The values are read in with the program inputs (see section 4).

logical variables The logical variables **quad** and **kick** are used to tell the `derivs` subroutine whether the muon is in a kicker region, a quadrupole region, or neither—the `derivs` subroutine needs to know whether or not to apply the fields appropriate to each of those regions. They are defined here in the `parameters` module so that the `derivs` subroutine and the main program will have access to the values. Each time the main program calls the `odeint` routine, it also sets the values of **quad** and **kick**. If more distinct regions are added, there can be more logical variables.

6 Kicker field

So far, the magnetic field between the kicker plates is modeled as completely uniform (with no fringe fields), entirely in the vertical direction, and with the time-dependence of an underdamped RLC circuit. The peak value of the magnetic field and the time when it occurs are read in with the program inputs (see section 4); R, L, and C for the circuit are defined in the `parameters` module (see section 5).

If the logical variable **kick** is set to true when `muon_trajectory_11` calls `odeint`, the `derivs` subroutine will include the kicker field in the differential equations (Equation 14); otherwise, the magnetic field will just be the dipole field of the storage ring. The kicker field itself is a function within `derivs.f90`. It is written as a function of x, y, and t, but right now there is no spatial dependence (it might be desirable to add some spatial dependence in the future).

The time-dependence of the current in an RLC circuit is given by

$$I \propto e^{-\alpha t} \sin(\omega t), \tag{15}$$

where $\alpha = R/2L$ is the strength of the damping, $\omega_0 = \sqrt{1/LC}$ is the natural frequency, and $\omega = \sqrt{\omega_0^2 - \alpha^2}$ is the frequency of the oscillation. In `derivs.f90`, this function is divided by its maximum value, so that it will have a peak value of one (it will later be multiplied by the input parameter **kickermagnitude**). In place of t, the equation has (t + **tzerotomax** - **kickermaxtime**); this merely sets the peak magnetic field at the desired time. **tzerotomax** is the time it takes the function to increase from zero to its maximum value; **kickermaxtime**, one of the input parameters, is the time at which we want the peak magnetic field to occur. Incidentally, I worried for a while about what would happen if a muon entered the kicker region *before* the kicker was turned on; I contemplated setting up some kind of conditional function to make sure **kickertimedep** would be zero in that situation. However, I determined that it wouldn't make any sense for the kicker magnet to be turned on that late; **tzerotomax** = 156 ns, and it only takes 150 ns for a muon to travel all the way around the storage ring—by rights, the kicker has to be turned on well before the muons enter the ring.

Finally, note that the kicker field is in the negative y-direction; it has to *oppose* the storage ring dipole field in order to have the desired effect.

7 Electrostatic quadrupole field

Just like the kicker's magnetic field, the quadrupoles' electric field is calculated by a function within the `derivs` subroutine. It is a function of x and y; there is no time dependence and no consideration of fringe fields at the ends of the quadrupoles.

I got my information about the quadrupole field from Ref. [7], Equation (19) and Table 5: an equation for a multipole expansion of the potential and a list of coefficients. The b_2 coefficient is the dominant quadrupole field, and it is the most important. It has an obvious effect on the trajectories of muons. I added in a couple other terms (the b_4 and b_6 coefficients), but didn't see much change to the trajectories. In principle, more terms could be added. It is simply a matter of adding terms to the electric field.

Note: I use the negatives of the values listed in Ref. [7] Table 5, because that table is for the storage of negative muons, and this experiment uses positive muons. Also, the values of the coefficients in the table are given in Volts; to get the actual a_n or b_n coefficient, you have to divide the value in the table by $(4.5\text{ cm})^n$; this is done in the `parameters` module, as well as converting to the appropriate program units.

8 Passing through the quadrupole plate

The muons enter the storage ring at a radial distance of 77 mm out from the desired orbit-path. The outer quadrupole plate is located at 50 mm. It turns out that the muons have to pass *through* the quadrupole plate as they travel through the first quadrant of the ring.

There are a number of steps that `muon_trajectory_11` takes to deal with this situation:

- The program calls `odeint` once for each element of the ring, so it needs to know the angle around the ring at which each element begins and ends. Normally that is straightforward, but in this case, the muon isn't between the quadrupole plates until it passes through the outer plate....and where exactly that happens depends on the initial position and momentum of the muon. So, the first thing the program does when it tracks a muon is calculate where the muon will hit the plate.

- Next, the program calls `odeint` to track the muon from the inflector to the point where it strikes the quadrupole plate. In principle, the coordinates at the Q1 plate could be calculated analytically, but it is simpler to let the computer do the tracking numerically, and makes the stored trajectory data consistent throughout the muon's time in the ring.

- Passing through the aluminum quadrupole plate scatters the muons; the program applies a gaussian-distributed scattering angle to each muon. The width of the distribution depends on how far the muon travels through the plate—which depends on the angle of approach to the plate. Thus, the program calculates the path length, based on the angle of approach, then calculates the width of the scattering-angle distribution (according to a formula in [6]), then alters the trajectory of the muon by a random scattering angle.

- After applying the random scattering angle, the tracking proceeds normally through the rest of the ring.

Here is how the program calculates the location around the ring where the muon will hit the quadrupole plate: under the influence of the storage ring's dipole field, the muon travels in a circle, but a circle that is off-center from the center of the storage ring (the whole point of the kicker magnet is to shift the circular trajectory so that it *is* centered on the ring).

Based on the initial position and momentum of the muon, I calculate the radius and center of its circular trajectory, and use that information to calculate the location where the muon's circular trajectory will intersect the quadrupole plate. Figure 1 shows the basic setup of the problem.

The radius of the muon's trajectory is determined by its momentum and the magnetic field of the storage ring. In my dimensionless units,

$$R = \frac{p_{\text{horizontal}}}{B}. \tag{16}$$

The small angle α is determined by the direction of the initial momentum,

$$\alpha = \arctan\left(\frac{p_{\text{radial}}}{p_{\text{azimuthal}}}\right). \tag{17}$$

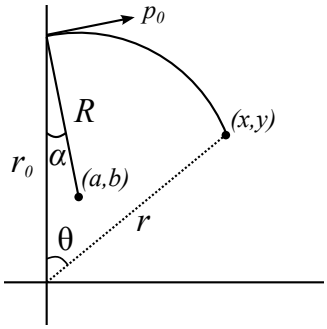

Figure 1: Geometry used to calculate where the muon strikes the quadrupole plate in its first turn around the storage ring. The origin is located at the center of the storage ring. The muon enters the ring a distance r_0 away from the center of the ring, with initial momentum p_0, and it travels in a circle of radius R centered on the point (a, b). Some time later, the muon is located at the point (x, y)—or, in polar coordinates, (r, θ). The muon hits the quadrupole plate when $r = r_q$, the radius of the outer quadrupole plate.

Thus, the center of the muon's circular trajectory is located at

$$a = R\sin\alpha = \frac{p_{\text{radial}}}{B} \tag{18}$$

$$b = r_0 - R\cos\alpha = r_0 - \frac{p_{\text{azimuthal}}}{B}. \tag{19}$$

Points on the circular trajectory satisfy

$$(x-a)^2 + (y-b)^2 = R^2, \tag{20}$$

which in polar coordinates is

$$(r\sin\theta - a)^2 + (r\cos\theta - b)^2 = R^2 \tag{21}$$

$$r^2 + a^2 + b^2 - 2r(a\sin\theta + b\cos\theta) = R^2 \tag{22}$$

$$r^2 + a^2 + b^2 - 2r\sqrt{a^2+b^2}\cos\left(\theta - \arctan\left(\frac{a}{b}\right)\right) = R^2. \tag{23}$$

Now we can solve for θ and set $r = r_q$, the radius of the outer quadrupole plate:

$$\theta = \arccos\left(\frac{r_q^2 + a^2 + b^2 - R^2}{2r_q\sqrt{a^2+b^2}}\right) + \arctan\left(\frac{a}{b}\right). \tag{24}$$

Note: this is the correct equation even if the initial radial momentum is negative; that just means that α and a are also negative.

I got my information about scattering from the Particle Data Group (Ref. [6], section 27, "*Passage of particles through matter*"). This article tells me that small-angle multiple scattering is roughly gaussian, with the width (θ_0) given by Ref. [6] Equation (27.14). This width depends on the velocity, momentum and charge number of the incident muon, as well as the thickness and radiation length of the target material. For the velocity and momentum of the muons, I used $\beta = 1$ (it's actually about 0.999) and $p = 29.3$ MeV/c (the "magic momentum"); even though each muon will have a slightly different momentum when it hits the quadrupole plate, those small variations shouldn't make an appreciable difference to the scattering width. The charge number is $z = 1$, because a muon has the same charge as an electron. I found the radiation length of aluminum (8.897 cm) in another section of the PDG document (Ref. [6], section 6, "*Atomic and nuclear properties of materials*").

That just leaves the thickness of the material. The outer Q1 plate is 0.1 mm thick (Ref. [7], Section 4.1.4), but the muons don't strike the plate head-on. They approach at such a shallow angle that their travel-length through the aluminum is on the order of 1 cm. `Muon_trajectory_11` calculates the path length for each muon as

$$\text{path length} = (\text{plate thickness})\frac{p_{\text{azimuthal}}}{p_{\text{radial}}}, \tag{25}$$

making use of a small-angle approximation. This approximation is justified, because for muons that successfully hit the quadrupole plate, $p_{\text{radial}}/p_{\text{azimuthal}}$ is on the order of 0.01.

Currently, `muon_trajectory_11` applies a scattering angle, but does not account for the displacement of the muon as it travels through the quadrupole plate; it treats the scattering as though it all happens at the point where the muon strikes the plate. It also does not account for the energy lost by the muon as it travels through the material. I would guess that the energy loss might be significant, but the displacement is probably not significant.

9 Program output

The `odeint` subroutine stores coordinate values in a module; the input parameter **dxsav** tells it how often (in radians) to save data points. After each call to `odeint`, `muon_trajectory_11` writes the trajectory information to a file. The output file has seven columns: turn number, radial position (x) in meters, vertical position (y) in meters, time in nanoseconds, radial momentum (p_x) in units of mc, vertical momentum (p_y) in units of mc, and the difference between the azimuthal momentum and the "magic momentum" ($p_z - 29.3$) in units of mc.

If you want the output to be in some other form, you will need to alter all the write statements in `muon_trajectory_11`; there is one write statement after each call to `odeint`, because a new call to `odeint` erases the information previously stored in the `ode_path` module.

There is a separate data file for each muon; they are named "muon1," "muon2," etc., and are saved in the directory specified in the input file.

10 Summary and remaining work

In summary, here is a list of things included in `muon_trajectory_11`:

- RLC time dependence for the kicker field (but no spacial dependence and no fringe fields).
- Electrostatic quadrupole fields, including a couple of higher multipoles (but not yet all the terms).
- Scattering when the muons pass through the outer Q1 plate (but no energy loss).

Here are things that still need to be done and ideas for future work with the muon tracking program:

- For every muon that ends up on a reasonable trajectory through the storage ring, there will be many muons that are lost from the ring. It will be very important for the tracking program to identify particles that are lost from the muon bunch by hitting the beam pipe walls or otherwise going out-of-bounds. `Muon_trajectory_11` currently does not include *any* such checks. This was the next thing on my agenda.
- Energy loss when the muons pass through the Q1 plate.
- More terms to the quadrupole fields
- Determine what the distributions of initial coordinates should look like when the muons come out of the inflector end.
- Programs to handle output data, calculate & display quantities of interest.

And various ideas for the improvement of the kicker:

- Locate a second kicker directly across the ring from the first.
- Adjust the timing and/or peak magnitude of the kicker pulse.
- Introduce a radial spacial-dependence to the kicker field.
- Adjust the shape of the kicker pulse so that it has a smaller magnitude but longer duration; attempt to deliver the kicker pulse over two cycles, not just one.

11 Acknowledgments

I am indebted to David Rubin, my adviser for this project. To the best of my knowledge, my work was funded by the NSF (though I do not know the grant number).

References

[1] New (g-2) Collaboration, "The new (g-2) experiment: a proposal to measure the muon anomalous magnetic moment to ±0.14 ppm precision," (2010).

[2] J. P. Miller, E. de Rafael, and B. L. Roberts, "Muon (g-2): experiment and theory," Reports on Progress in Physics **70**, 795 (2007).

[3] M. Metcalf and J. Reid, *Fortran 90 Explained* (Oxford University Press, 1990).

[4] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in Fortran* (Cambridge University Press, 1996), 2nd ed.

[5] E. Efstathiadis et al., "A fast non-ferric kicker for the muon (g-2) experiment," Nuclear Instruments & Methods in Physics Research A **496**, 8 (2002).

[6] K. Nakamura et al., "Review of Particle Physics," J. Phys. G **37** (2010).

[7] Y. K. Semertzidis et al., "The Brookhaven muon (g-2) storage ring high voltage quadrupoles," Nuclear Instruments & Methods in Physics Research A **503**, 458 (2003).