Personal notes on MAD-X

D. Gamba

Jul 2014

1 Verify MAD-X

For my simulations I assume that MAD-X TWISS is able to compute with good accuracy the beam coordinates and Twiss functions along a beam line when specifying an initial DELTAP.

To verify that the non linearity in $\Delta p/p_0$ is computed correctly one can simulate a simple line made of a single element and a single drift afterwards. If one considers a single kicker, and makes many TWISS with a wide range of different initial DELTAP values, one will see a non linear dependancy on $\Delta p/p_0$ in the final beam position. In such a simple system it is possible to analytically calculate all the higher orders terms:

$$x_{end} = \alpha \left(1 - \frac{\Delta p}{p_0} + \left(\frac{\Delta p}{p_0}\right)^2 - \left(\frac{\Delta p}{p_0}\right)^3 + ...\right) L \tag{1}$$

$\alpha :=$ corrector kick;

$L :=$ drift length.

In Fig. 1 a simulation of this system is shown. To be notice that the simulated interval in energy error is very wide and far from any practical use, but the convergence of the theoretical higher order terms to the simulation is a proof of the idea.

The same can be done simulating similar conditions with a quadrupole where the beam enters with an initial offset or because of quadrupole misalignment (Figure 2). In all the cases MAD-X seems to do what is expected.

Without proving it, one could expect MAD-X is also giving the correct result after a sequence of elements and not only after single one.

If one wants to simulate also longitudinal dependency on $\Delta p/p_0$, one should take in consideration the canonical variable T and PT. According to the documentation of MAD-X the T variable is the delta time of flight of the of the simulated beam respect to the designed beam (all zero initial conditions), multiplied by the speed of light c. It is not to me perfectly clear the correct definition of the general coordinate system, but by running TWISS on a simple drift element, one can observe that:

- Starting with all initial coordinates equal to zero, after the drift all the coordinate are zero as well.

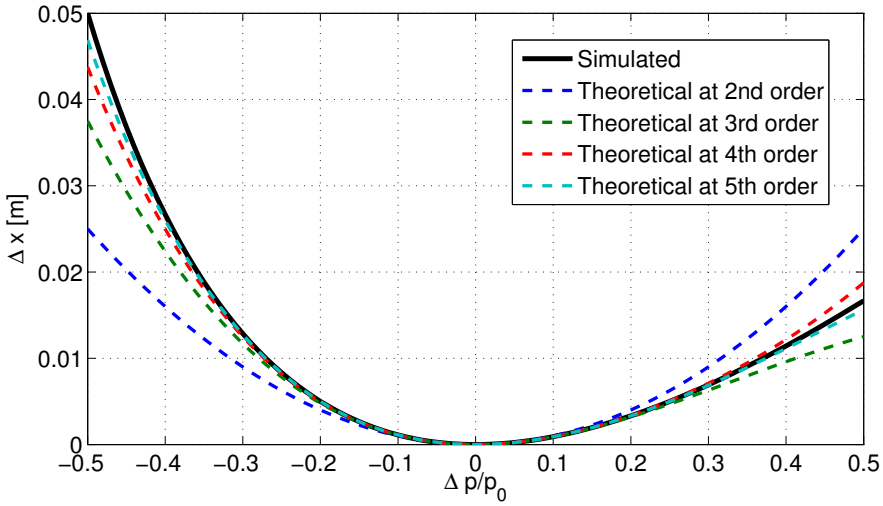

Figure 1: Non linear dependency on $\Delta p/p_0$ of a corrector before a 1 m long drift. The black line is simulated by means of MAD-X Twiss calculations with different initial DELTAP values. The dashed lines are calculated according eq. 1 truncated at different orders.

- Starting with an initial PX value different from zero, after the drift one gets:

$$\mathrm{X}_{end} = \mathrm{PX} \times L \tag{2}$$

$$\mathrm{T}_{end} = \frac{\sqrt{\mathrm{X}_{end}^2 + L^2}}{\beta_0} \tag{3}$$

  where L is the drift length and β_0 is the nominal relativistic β.

- Starting with an initial DELTAP value, after the drift one gets a change on the coordinate T only that is approximately $\mathrm{T}_{end} = L \times \mathrm{DELTAP}/\beta_0^2\gamma_0^2$. To be noticed that one gets PT_{end} equal to zero.

From this I understand that MAD-X coordinates are all referred to the ideal orbit (i.e. all initial values equal to zero), except PT that is referred to the particle with momentum corrected by the given DELTAP, and following the documentation it should be:

$$\mathrm{T} := -\Delta t\, \mathrm{c} \tag{4}$$

$$\mathrm{PT} := \frac{E - E_s}{p_s \mathrm{c}} = \frac{\gamma - \gamma_s}{\gamma_s \beta_s} \tag{5}$$

where Δt is the difference in time of flight of the considered particle respect to the ideal particle (i.e. particle with all initial conditions equal zero, DELTAP included) and E is the energy of the particle. Following the notation found in the code and in

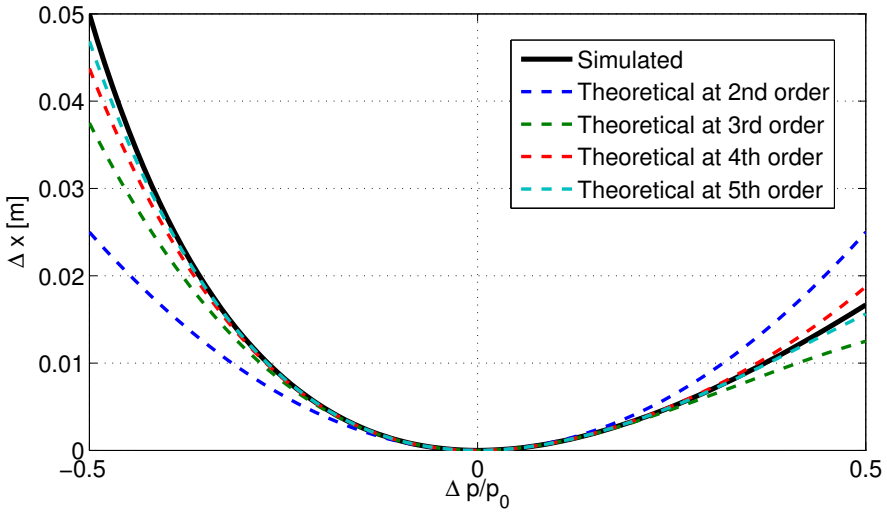

Figure 2: Non linear dependency on $\Delta p/p_0$ of a misaligned quadrupole before a 1 m long drift. The black line is simulated by means of MAD-X Twiss calculations with different initial DELTAP values. The dashed lines are calculated according eq. 1 truncated at different orders. The strength of the quadrupole is set such that a 1 mm misalignment gives a kick to the beam of 0.1 rad.

the documentation we add a subscript 0 to the parameters of the ideal particle, and a subscript s to the parameters of the reference particle, i.e. the particle tracked according to the initial conditions of TWISS, in particular with the specified DELTAP $:= \delta_s$.

It is here useful to remind here few equations from special relativity:

$$E = \gamma m_0 \mathrm{c}^2 \tag{6}$$

$$p = \gamma \beta m_0 \mathrm{c} \tag{7}$$

$$\gamma := \sqrt{\frac{1}{1-\beta^2}} \tag{8}$$

$$\beta := \frac{V}{\mathrm{c}} \tag{9}$$

and so one can derive these relations:

$$\beta\gamma = \frac{p}{m_0 \mathrm{c}} \tag{10}$$

$$\beta = \frac{\beta\gamma}{\sqrt{1+(\beta\gamma)^2}} \tag{11}$$

$$\gamma = \sqrt{1+(\beta\gamma)^2} \tag{12}$$

and defining $p = p_0(1 + \Delta p/p_0)$:

$$\begin{aligned}\beta\gamma &= \frac{p_0}{m_0 \mathrm{c}}\left(1 + \frac{\Delta p}{p_0}\right) \\ &= \beta_0\gamma_0\left(1 + \frac{\Delta p}{p_0}\right) \qquad (13)\end{aligned}$$

$$\beta = \frac{\beta_0\gamma_0}{\sqrt{\frac{1}{(1+\Delta p/p_0)^2} + (\beta_0\gamma_0)^2}} \qquad (14)$$

and one can use the same relationship to refer p to p_s.

One may want to refer the MAD-X coordinate PT to the more commonly used $\Delta p/p$:

$$\begin{aligned}\frac{\Delta p}{p_s} &= \frac{\gamma\beta - \gamma_s\beta_s}{\gamma_s\beta_s} \\ &= \frac{\gamma\beta/\beta_s - \gamma_s}{\gamma_s\beta_s}\beta_s \\ &\approx \mathrm{PT}\beta_s \qquad (\beta/\beta_s \approx 1) \qquad (15)\end{aligned}$$

Nevertheless it has to be noticed that in MAD-X documentation often it is assumed:

$$\mathrm{PT} \approx \frac{\Delta p}{p_s}\beta_s \qquad (16)$$

that is agains eq. 16, but it looks like it is widely used in the code it is analysed afterwards. One can verify eq. 16 expanding the two terms:

$$\begin{aligned}\frac{E - E_s}{p_s\mathrm{c}} &\overset{?}{\approx} \frac{p - p_s}{p_s}\beta_s \\ E - E_s &\overset{?}{\approx} \beta_s p\mathrm{c} - \beta_s p_s\mathrm{c} \\ E - E_s &\overset{?}{\approx} E\beta_s/\beta - E_s\end{aligned}$$

that is verified in the approximation of $\beta_s/\beta \approx 1$. Both assumptions seems to be reasonable, but one my want too check if there is one that is better than the other. Figure 3 shows that the approximation done by MAD-X documentation better fits the real value of PT for a wide range of energies for a reasonable value of $\Delta p/p_s$.

Given the assumptions above, if one considers just a drift of 1 m length, and one runs a TWISS with zero initial conditions, but different DELTAPs, one should only see the small[1] effect induced by the different simulated velocities. By MAD-X conventions one can write:

$$\begin{aligned}\mathrm{T} := -\Delta t\,\mathrm{c} &= -\frac{L}{\beta_0}\left(1 - \frac{1}{1 + \Delta\beta/\beta_0}\right) \\ &= -\frac{L}{\beta_0}\left(\frac{\Delta\beta/\beta_0}{1 + \Delta\beta/\beta_0}\right) \qquad (17)\end{aligned}$$

[1]Depending on the energy of the beam it should be close to zero for ultra-relativistic energies.

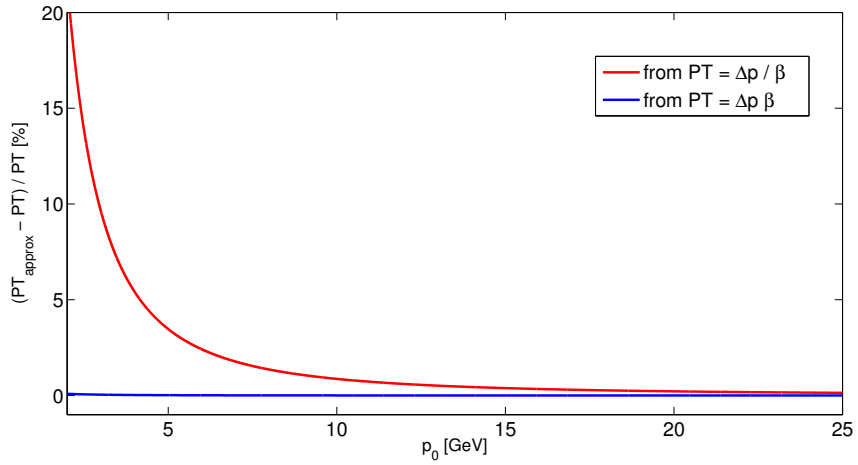

Figure 3: Relative error in computing PT from approximations of eq. 15 and 16 along the energy range of the PS, supposing a constant $\Delta p/p_s = 1\%$.

where $\Delta\, t$ is the delta time of flight of the beam in a drift long L respect to a beam with design momentum.

From eq. 14 one can compute $\Delta\beta/\beta_0$:

$$\frac{\Delta\beta}{\beta_0} = \frac{\gamma_0}{\sqrt{\frac{1}{(1+\Delta p/p_0)^2} + (\beta_0\gamma_0)^2}} - 1 \tag{18}$$

By joining eq. 17 and eq. 18 one can finally write

$$\begin{aligned} \mathrm{T} &= -\frac{L}{\beta_0}\left(1 - \frac{\sqrt{\frac{1}{(1+\Delta p/p_0)^2} + (\beta_0\gamma_0)^2}}{\gamma_0}\right) \\ &= -L\left(\frac{1}{\beta_0} - \sqrt{\frac{1}{(\beta_0\gamma_0)^2(1+\Delta p/p_0)^2} + 1}\right) \end{aligned} \tag{19}$$

Now one can Taylor expand eq. 19 in $\Delta p/p_0$, and at first order one finds

$$\mathrm{T} \approx -L\frac{1}{\beta_0\gamma_0^2}\frac{\Delta p}{p_0} \tag{20}$$

At this point by eq. eq. 19 and eq. 20 and my current understanding of the MAD-X coordinates meaning one should expect that for the sequence under consideration (i.e. a simple drift of length L) a TWISS with initial only DELTAP value different from zero

should give at the end:

$$\mathrm{T}_{end} = -L\left(\frac{1}{\beta_0} - \sqrt{\frac{1}{(\beta_s\gamma_s)^2}+1}\right) \tag{21}$$

$$\mathrm{R56}_{end} \approx -\frac{L}{\beta_s^2\gamma_s^2}. \tag{22}$$

Equation 22 is only an approximation using eq. 16.

By means of simple simulations showed in fig. 4 it is however not true. In particular it looks like MAD-X is able to correctly compute R56 (and eventually the second order T566), but not T for the simulations varying DELTAP.

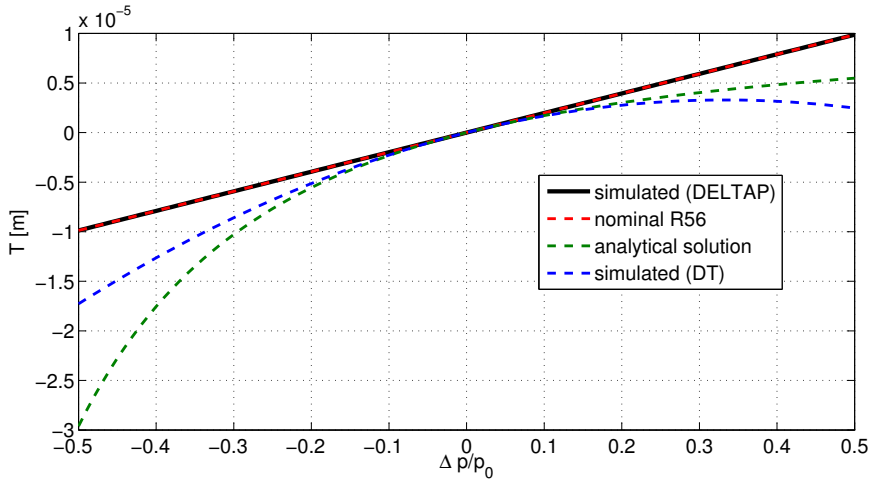

Figure 4: Non linear dependency on $\Delta p/p_0$ of the canonical variable T in 1 m long drift. The black solid line is simulated by means of MAD-X Twiss calculations with different initial DELTAP values. The dashed green line is what one would expect by eq. 19. The dashed red line, perfectly on top of the black one, is the linear approximation using the R56 value given by MAD-X for the DELTAP = 0 case. Finally the dashed blue line is by simulating different initial PT values.

1.1 Looking at the source code

By looking at the source code (version 5.02.02), when running a TWISS these are the relevant steps done by the code. The file *mad_twiss.c* is the C interface between the command line and the *twiss.f90* where is implemented most of the physics. The main C function is *pro_twiss(void)*:

1. It loads all the parameters from the TWISS command line.
2. At line 927 it loads all the DELTAP values given in the TWISS command.

3. At line 929 it calls the C function *adjust_beam()*. This sets the values of relativistic parameters for the designed energy (e.g. *beta*, *gamma*, ...), as well the length of the sequence: *circ*.

4. At line 931 it calls the FORTRAN function *tmrefe* that (should) computes the transfer matrix for the whole sequence for the designed-ideal beam (i.e. zero initial conditions, no imperfections).

5. At line 1000 it starts cycling on all DELTAP values.

6. At the first iteration over the various DELTAPs, the variable *oneturnmat*, i.e. the linear transfer matrix of the whole sequence, is computed for the design case only (all initial conditions equal to zero).

   At line 1028 it calls the *adjust_probe(DELTAP[i])* that should adjust, among other things, the new reference relativistic parameters (e.g. *betas*, *gammas*, ...). It is important to see what happens inside the function *adjust_probe*.

   (a) This function gets the nominal relativistic parameters β_0, γ_0 and the total length of the sequence (*circ*). It also uses the *oneturnmat* and the *disp0* currently stored in memory.

   (b) MAD-X wants to adjust the revolution frequency (*freq0*) for the probe beam, i.e. the beam with the specified DELTAP. Nevertheless it doesn't have the total path length of the beam path for the given initial conditions, but only the sequence length. So it is trying to adjust the beam path as linear dependency on DELTAP respect to the design machine and the given initial dispersion[2], that should be stored in the variable *disp0*. This is supposed to be transported in the sequence like a particle with nominal momentum p_0. So the code computes:

$$\mathrm{ds} := \mathrm{oneturnmat}[5, 1:4] \times \mathrm{disp0} + \mathrm{oneturnmat}[5, 6]. \tag{23}$$

   At this stage *ds* is the factor such that T = ds × PT for a particle with designed energy plus an energy error given by PT. The revolution frequency can be defined as

$$\begin{aligned} \mathrm{freq0} &= \frac{\mathrm{c}\,\beta_0}{L_{path}} \\ &= \frac{\mathrm{c}\,\beta_0}{\mathrm{circ} - \mathrm{T}\beta_0} \\ &\approx \frac{\mathrm{c}\,\beta_0}{\mathrm{circ} - \mathrm{ds} \times \mathrm{DELTAP}\beta_0^2} \end{aligned} \tag{24}$$

   using the approximation of eq. 16.

[2] I didn't find where it is initialised, so it might be zero always zero here.

(c) Afterwords the adjusted relativistic parameters are computed correctly as

$$\eta_s := \gamma_s\beta_s = \gamma_0\beta_0\,(1 + \text{DELTAP}) \tag{25}$$

$$\gamma_s = \sqrt{1+\eta_s^2} \tag{26}$$

$$\beta_s = \frac{\eta_s}{\gamma_s}. \tag{27}$$

(d) At this point, similar to the *freq0*, MAD-X computes an approximation of the phase slippage factor defined as:

$$\text{alfa} := \frac{1}{\gamma^2} - \frac{\Delta L_{path}}{L_{path}} \tag{28}$$

where ΔL_{path} should be the differential path length of a full turn in the machine due to a momentum error $\Delta p/p$. Normally one would compute a linear approximation compatible with *freq0* calculated before:

$$\text{alfa} = \frac{1}{\gamma_0^2} - \frac{\beta_0^2\,\text{ds} \times \text{DELTAP}}{\text{circ}} \tag{29}$$

but MAD-X instead uses

$$\text{alfa} = \frac{1}{\gamma_s^2} - \frac{\beta_s^2\,\text{ds} \times \text{DELTAP}}{\text{circ}}. \tag{30}$$

Equation 30 is using the assumption of eq. 16, and it uses γ_s and β_s that may not be compatible with the assumptions done for *freq0*.

(e) For my purpose neither *freq0*, nor *alfa* should be important, but now it is also computed the parameter *dtbyds*.

This should be meant to be the term in parenthesis of eq. 19, in the sense that multiplied by a path length, it should give kick it has to be applied on the coordinate T due to a reference relativistic $\beta = \beta_s \neq \beta_0$. This value should then be zero if $\beta_s = \beta_0$, i.e. DELTAP = 0. In MAD-X it is calculated as:

$$\text{dtbyds} = -\frac{\beta_s\,\text{ds} \times \text{DELTAP}}{\text{circ}}. \tag{31}$$

Again eq. 31 looks reasonable using the approximation of eq. 16, but it doesn't serve the scope it is meant for. It shows in fact only a linear dependency on DELTAP, and non linear dependency on β_s that is anyway always very close to one for our typical energies, and so negligible. One has to remind that *ds* is calculated according to the nominal energy case, and so it does not depend on DELTAP.

In our specific case of a sequence with only one drift long L, starting with zero initial dispersion and with *oneturnmat* calculated only for the design

beam, the only non zero term in the computation of *ds* is the element *oneturnmat[5,6]*, and so

$$\mathrm{ds} = \frac{L}{(\beta_0\gamma_0)^2} \tag{32}$$

that is coming from the transfer map of a drift at designed energy in the FORTRAN function *tmdrf0* (this is used again afterwards, see below). Equation 32 together with eq. 31 and considering that for our case it is circ = L, one gets:

$$\mathrm{dtbyds} = -\frac{1}{(\beta_0\gamma_0)^2}\beta_s\,\mathrm{DELTAP}. \tag{33}$$

that looks again compatible with a linear approximation always keeping in mind the assumption of eq. 16.

Nevertheless the value of *dtbyds* doesn't need to be computed from the total path length of the sequence, but is a well defined quantity at the particular energy like we saw before in eq. 19. A better value should then be

$$\mathrm{dtbyds} := \left(\frac{1}{\beta_0} - \sqrt{\frac{1}{(\beta_s\gamma_s)^2} + 1}\right). \tag{34}$$

Inside eq. 34 is in fact explicit the non linear dependency on DELTAP by the presence of the term γ_s that quickly changes for different DELTAP.

7. At line 1033 it calls the FORTRAN function *twiss* that actually does the computation of the full TWISS table. Here the main steps are:

   (a) For an open line, it gets the initial conditions (*orbit0*) from the C interface. For a closed ring it computes finds the closed solution using the function *tmclor*.

   (b) It executes the real TWISS computation using the function *tmfrst*.

   (c) The function *tmfrst* first gets the probe relativistic parameters (i.e. *betas*, *gammas*, ...).

   (d) Afterwards by cycling on each node of the given sequence, it gets the transfer map for each node using the function *tmmap*. For the particular case of a drift, the main function that generates the map is *tmdrf0*. Here is computed:

$$\mathrm{ek}(5) = \mathrm{dl} \times \mathrm{dtbyds} \tag{35}$$

$$\mathrm{re}(5,6) = \frac{\mathrm{dl}}{(\beta_s\gamma_s)^2} \tag{36}$$

   where dl is the drift length, meanwhile dtbyds, β_s and γ_s are the one generated before by *adjust_probe*.

   Once again the re(5,6) term is only compatible with assumption of eq. 16.

(e) Each map generator function (e.g. *tmdrf0*) is also responsible to advance the *orbit* using the generated map at first and second order by using the function *tmtrak*.

8. Now the control is back to the C interface. If the user has provided a range of DELTAPs, the code goes back to the code described from item 6. At this stage the *oneturnmap* is computed for the the current DELTAP. This means that it is not anymore the designed one, and in the calculation of *freq0*, *alfa*, *dtbyds* it may give a different result![3]

9. If the cycle over all the provided DELTAPs is done, then the final TWISS table is populated and delivered to the user.

[3]Not yet verified.