Methodical Accelerator Design
Bugs review and fixes

Laurent Deniau
CERN-BE/ABP

4th March 2016

Laurent Deniau, CERN BE/ABP, 1211 Geneva 23, laurent.deniau@cern.ch

Context

- Bastian reported unexpected changes in chromaticity between Twisses #385
  - ➡ FCC-ee studies have RF activated with harmon defined
- Ghislain simplified the examples and refined the definition of the problem:
  - ➡ repeated Twiss commands with deltap=0 give the same result
  - ➡ repeated Twiss commands with deltap≠0 give different results
  - ➡ subsequent Twiss commands with deltap≠0 converge to some results
- Laurent tracked the problem in the code
  - ➡ found few different correlated bugs
  - ➡ wrote a memo to save the information and to propose fixes

- Purpose of this meeting is to explain the bug(s) and the fixes
  - ➡ provide quick fixes for users (e.g. macros)
  - ➡ provide long-term fixes in the application (e.g. next release)
  - ➡ affect twiss, embedded_twiss, track, dynap, emit, touschek, IBS, plot, PTC twiss, …

Laurent Deniau, CERN BE/ABP, 1211 Geneva 23, laurent.deniau@cern.ch

Problem

```
call, file = "fcc-ee_lin-rt9060-1m2mm.str"; ! Default magnet strengths
call, file = "fcc-ee_lin-elements.madx";    ! Element definitions
call, file = "fcc-ee_lin.seq";              ! Sequence definitions
beam, particle=electron, energy=175, npart=9e12, radiate=false;
use, sequence=FCCeeRacetrack;
dp = 1e-5;
twiss, deltap = dp; ! TWISS-1
twiss, deltap = dp; ! TWISS-2
twiss, deltap = dp; ! TWISS-3
```

	TWISS-1	TWISS-2	TWISS-3
! init			
ds	**-0.5466365238079542 44**	-0.5**06917710659540388**	-0.506917710659**382292**
f0	0.0029979245798233416	0.00299792457983**52487**	**0.0029979245798352487**
alfa	5.4663737642266372e-06	5.**0691856327458846**e-06	5.06918563274**4304**e-06
gammatr	427.71091985215514	**444.1512639497102**	444.151263949**7943**
! twiss			
orbit5	**-2.7331815849983363e-11**	-2.**5332995020550225**e-11	-2.533299509**0933369**e-11
alfa	5.0693603411216544e-06	5.0693**513300685796**e-06	**5.0693513300685796e-06**
gammatr	444.14361035938731	444.14**400510415908**	**444.14400510415908**
q1	**501.07417631102908**	501.074**59944213031**	501.0745994421**2957**
q2	**335.13136024163310**	335.131**99202617392**	335.131992026**17568**
dq1	**-539.58670522331965**	-539.58**202045356052**	-539.582020453**57302**
dq2	**-801.16036181961704**	-80**0.30808584169347**	-800.3080858**484899**

Twiss is supposed to be deterministic…

Laurent Deniau, CERN BE/ABP, 1211 Geneva 23, laurent.deniau@cern.ch

Twiss pre-initialization principle

Original

```
pro_twiss()  // ~ 400 lines
{
  …

// clear global data
  zero_double(orbit0, 6);
  zero_double(oneturnmat, 6*6);

// setup beam parameters and create a local clone
  adjust_beam();
  probe_beam = clone_command(current_beam);

// prepare one-turn-map for next probe adjustment
  tmrefe_(oneturnmat); // ONE TURN MAP (freq=0)
  …
  foreach deltap do {
    if (chrom_flag) {
      adjust_probe(deltap+1e-6);
      adjust_rfc();
      twiss_(oneturnmat, disp0, …); // TWISS
    }
    …
// update probe parameters to dp
    adjust_probe(deltap);
// update RF->freq = probe->freq0 * RF->harmon
    adjust_rfc();

    twiss_(oneturnmat, disp0, …); // TWISS
    …
  }
  …
  probe_beam = delete_command(probe_beam);
}
```

```
adjust_beam()  // ~ 30 lines
{
// load beta, gamma from current_beam

  circ = current_sequ->length;
  alfa = one / (gamma*gamma);
  freq0 = (beta*clight*ten_m_6) / circ;

// save beta, gamma and circ to current_beam
}
```

```
adjust_probe(deltap)  // ~ 80 lines
{
// load circ, beta, gamma from current_beam

  ds = oneturnmat[4 + 6*5]; // R(5,6)
  for (int j=0; j < 4; j++)
    ds += oneturnmat[4 + 6*j] * disp0[j]; // R(1:4,6) * D(1:4)

// step 1
  tmp = - beta * beta * ds / circ;
  freq0 = (beta*clight*ten_m_6) / (circ*(one + tmp*deltap));
  etas = beta * gamma * (one + deltap);
  gammas = sqrt(one + etas * etas);
  betas = etas / gammas;
// step 2
  tmp = - betas * betas * ds / circ;
  alfa = one / (gammas * gammas) + tmp;
  dtbyds = deltap * tmp / betas;

// save freq0, alfa, beta, gamma, … to probe_beam
}
```

```
adjust_rfc()  // ~ 20 lines
{
  foreach rfcavity in current_sequ do {
    if harmon is defined do {
      rfcavity->freq = freq0 * rfcavity->harmon;
    }
  }
}
```

Laurent Deniau, CERN BE/ABP, 1211 Geneva 23, laurent.deniau@cern.ch

Inconsistencies

RF cavivity with freq=0 (uninitialized)

$$\omega = \frac{2\pi}{c} f_{\mathrm{rf}}$$
$$A = \frac{V_{\mathrm{rf}}}{pc(1+\delta_p)}$$
$$\phi = 2\pi \mathrm{lag} - \omega \Delta s$$
$$c_0 = A * sin(\phi)$$
$$c_1 = -A * cos(\phi) * \omega$$
$$c_2 = -A * sin(\phi) * \omega^2/2$$
$$ek(6) = c_0 - c_1 \Delta s + c_2 (\Delta s)^2$$
$$re(6,5) = c_1 - 2 c_2 \Delta s$$

$$f_{\mathrm{rf}} = 0 \quad ; \quad \mathrm{lag} = 0.5$$
$$\Rightarrow \quad \omega = 0 \quad \Rightarrow \quad \phi = \pi \quad \Rightarrow \quad c_0 = c_1 = c_2 = 0$$

behaves like a dead space

Wrong oneturnmap and dispersion

step 1: collect input

$$ds = R_{5,6} + \sum_{i=1}^{4} R_{i,6} D_i$$
$$\xi = -\beta^2 \frac{ds}{L}$$

step 2: update parameters

$$f_0 = \frac{\beta c}{L(1+\xi\delta_p)}$$ ***wrong if deltap≠0***

$$\eta = \beta\gamma(1+\delta_p)$$
$$\gamma = \sqrt{1+\eta^2}$$
$$\beta = \eta/\gamma$$

step 3: update dependencies

$$\xi = -\beta^2 \frac{ds}{L}$$
$$\alpha_c = \frac{1}{\beta^2} + \xi$$ ***wrong***

$$\mathrm{dtbyds} = \frac{\xi\delta_p}{\beta}$$ ***wrong if deltap≠0***

Laurent Deniau, CERN BE/ABP, 1211 Geneva 23, laurent.deniau@cern.ch

Problems & Fixes for Users — short term

- BUG1: RF frequencies are not initialized before computing the one turn map
  - ➡ FIX: discard first Twiss in your script.
- BUG2: Revolution frequency **freq0** is computed from « unstable » recursive dependencies (oneturnmat and disp0)
  - ➡ FIX: run few Twisses each time you change **RF** parameters, **beam** parameters or **deltap** (2 extra Twiss are enough for FCC, 0 extra Twiss is enough for LHC).
- BUG3: **ds** is computed from the previously computed oneturnmat and disp0, which depends on deltap
  - ➡ FIX: avoid **chrom** and ranges of **deltap**.
- BUG4: disp0 vector is not cleared
  - ➡ FIX: avoid starting at large dispersion.
- BUG5: RF multipoles and Crab cavities are not registered as RF cavities in the sequence
  - ➡ FIX: don't use RF multipoles or Crab cavities as RF cavities or set their **freq**.

```
pro_twiss()  // ~ 400 lines
{
  …

// clear global data
  zero_double(orbit0, 6);
  zero_double(oneturnmat, 6*6);

// setup beam parameters and create a local clone
  adjust_beam();
  probe_beam = clone_command(current_beam);

// prepare one-turn-map for next probe adjustment
  tmrefe_(oneturnmat); // ONE TURN MAP
  …
  foreach deltap do {
    if (chrom_flag) {
      adjust_probe(deltap+1e-6);
      adjust_rfc();
      twiss_(oneturnmat, disp0, …); // TWISS
    }
    …
// update probe parameters to dp
    adjust_probe(deltap);
// update RF->freq = probe->freq0 * RF->harmon
    adjust_rfc();

    twiss_(oneturnmat, disp0, …); // TWISS
    …
  }
  …
  probe_beam = delete_command(probe_beam);
}
```

Laurent Deniau, CERN BE/ABP, 1211 Geneva 23, laurent.deniau@cern.ch

Problem revisited

! **dp=0**	TWISS-1	TWISS-5	TWISS-6
! init			
ds	**-0.5466365238079542**44	**-0.5069177106593822**92	-0.5069177106**57431964**
f0	0.0029979245799872187	0.0029979245799872187	0.0029979245799872187
alfa	5.4663737643971585e-06	5.0691856329148262e-06	5.0691856328**953232**e-06
gammatr	427.7109198454840	444.15126394230901	444.15126394316343
! twiss			
orbit5	**1.0481081330692339e-17**	**1.0481081330692339e-17**	**1.0481081330692339e-17**
alfa	5.069185632895308e-06	5.069185632895308e-06	5.069185632895308e-06
gammatr	444.15126394316411	444.15126394316411	444.15126394316411
q1	**501.07999954346565**	501.07**999954346565**	**501.07999954346565**
q2	**335.13999974047073**	335.13**999974047073**	**335.13999974047073**
dq1	-539.53614075684948	-539.53614075684948	-539.53614075684948
dq2	-790.49304349398938	-790.49304349398938	-790.49304349398938

! **dp=1e-5**	TWISS-2	TWISS-3	TWISS-4
! init			
ds	-0.5**069177106574319**64	-0.5069177106**593822**92	**-0.5069177106593822**92
f0	0.0029979245798352487	0.0029979245798352487	0.0029979245798352487
alfa	5.0691856327248011e-06	5.069185632744304e-06	5.069185632744304e-06
gammatr	444.15126395063385	444.15126394977943	444.15126394977943
! twiss			
orbit5	**-2.5332994929758881e-11**	-2.533299**5090933369**e-11	**-2.5332995090933369e-11**
alfa	5.0693513300685855e-06	5.0693513300685796e-06	5.0693513300685796e-06
gammatr	444.14400510415879	444.14400510415908	444.14400510415908
q1	501.07**459944213088**	501.0745994421**2957**	**501.07459944212957**
q2	335.13**199202617460**	335.13199202617**568**	**335.13199202617568**
dq1	-539.58202045356097	-539.58202045357302	-539.58202045357302
dq2	-800.30808584179238	-800.30808584184899	-800.30808584184899

Twiss is supposed to be deterministic…

Laurent Deniau, CERN BE/ABP, 1211 Geneva 23, laurent.deniau@cern.ch

Twiss pre-initialization Fix 1 — next release

Original

```
pro_twiss()  // ~ 400 lines
{
  …
  zero_double(orbit0, 6);
  zero_double(oneturnmat, 6*6);

  adjust_beam();
  probe_beam = clone_command(current_beam);
  tmrefe_(oneturnmat); // ONE TURN MAP
  …

  foreach deltap do {
    if (chrom_flag) {
      adjust_probe(deltap+1e-6);
      adjust_rfc();
      twiss_(oneturnmat, disp0, …); // TWISS
    }
    …
    adjust_probe(deltap);
    adjust_rfc();
    twiss_(oneturnmat, disp0, …); // TWISS
    …
  }
  …
  probe_beam = delete_command(probe_beam);
}
```

Update 1

```
pro_twiss()  // ~ 400 lines
{
  …
  zero_double(orbit0, 6);
  zero_double(disp0, 6);
  zero_double(oneturnmat, 6*6);

  adjust_beam();
  probe_beam = clone_command(current_beam);
  adjust_rfc(); // init RF
  tmrefe_(oneturnmat); // ONE TURN MAP
  …

  foreach deltap do {
    if (chrom_flag) {
      adjust_probe(deltap+1e-6);
      adjust_rfc();
      twiss_(oneturnmat, disp0, …); // TWISS
    }
    …
    adjust_probe(deltap);
    adjust_rfc();
    twiss_(oneturnmat, disp0, …); // TWISS
    …
  }
  …
  probe_beam = delete_command(probe_beam);
}
```

RF frequency initialization, clearing disp0 vector

Laurent Deniau, CERN BE/ABP, 1211 Geneva 23, laurent.deniau@cern.ch

Twiss pre-initialization Fix 2 — NR or NNR

Update 1

```
pro_twiss()  // ~ 400 lines
{
  …
  zero_double(orbit0, 6);
  zero_double(disp0, 6);
  zero_double(oneturnmat, 6*6);

  adjust_beam();
  probe_beam = clone_command(current_beam);
  adjust_rfc(); // init RF
  tmrefe_(oneturnmat); // ONE TURN MAP
  …

  foreach deltap do {
    if (chrom_flag) {
      adjust_probe(deltap+1e-6);
      adjust_rfc();
      twiss_(oneturnmat, disp0, …); // TWISS
    }
    …
    adjust_probe(deltap);
    adjust_rfc();
    twiss_(oneturnmat, disp0, …); // TWISS
    …
  }
  …
  probe_beam = delete_command(probe_beam);
}
```

Update 2

```
pro_twiss()  // ~ 400 lines
{
  …
  zero_double(orbit0, 6);
  zero_double(disp0, 6);
  zero_double(oneturnmat, 6*6);

  adjust_beam(); // compute f0
  probe_beam = clone_command(current_beam);
  adjust_rfc(); // init RF
  tmrefe_(oneturnmat); // ONE TURN MAP
  twcpin_(oneturnmat,disp0,r0mat,&error);
  …

  foreach deltap do {
    if (chrom_flag) {
      adjust_probe(deltap+1e-6);
      adjust_rfc();
      twiss_(oneturnmat, disp0, …); // TWISS
    }
    …
    adjust_probe(deltap);
    adjust_rfc();
    twiss_(oneturnmat, disp0, …); // TWISS
    …
  }
  …
  probe_beam = delete_command(probe_beam);
}
```

compute disp0 before ds and freq0

Laurent Deniau, CERN BE/ABP, 1211 Geneva 23, laurent.deniau@cern.ch

Twiss pre-initialization Fix 3 — NR or NNR

Update 2

```
pro_twiss()  // ~ 400 lines
{
  …
  zero_double(orbit0, 6);
  zero_double(disp0, 6);
  zero_double(oneturnmat, 6*6);

  adjust_beam();
  probe_beam = clone_command(current_beam);
  adjust_rfc(); // init RF
  tmrefe_(oneturnmat); // ONE TURN MAP
  twcpin_(oneturnmat,disp0,r0mat,&error);
  …
  foreach deltap do {
    if (chrom_flag) {
      adjust_probe(deltap+1e-6);
      adjust_rfc();
      twiss_(oneturnmat, disp0, …); // TWISS
    }
    …
    adjust_probe(deltap);
    adjust_rfc();
    twiss_(oneturnmat, disp0, …); // TWISS
    …
  }
  …
  probe_beam = delete_command(probe_beam);
}
```

solve the fix point freq0 vs. ds

Update 3

```
pro_twiss()  // ~ 400 lines
{
  …
  zero_double(orbit0, 6);
  zero_double(disp0, 6);
  zero_double(oneturnmat, 6*6);

  adjust_beam();
  probe_beam = clone_command(current_beam);
  adjust_rfc(); // init RF
  …
  foreach deltap do {
    if (chrom_flag) {
      adjust_probe_fix_point(deltap+1e-6);
      twiss_(oneturnmat, disp0, …); // TWISS
    }
    …
    adjust_probe_fix_point(deltap);
    twiss_(oneturnmat, disp0, …); // TWISS
    …
  }
  …
  probe_beam = delete_command(probe_beam);
}
```

```
adjust_probe_fix_point(deltap)
{
  do { err0=0; // ds error
    tmrefe_(oneturnmat);
    twcpin_(oneturnmat,disp0,r0mat,&error);
    adjust_probe(deltap);
    adjust_rfc();
    for (int i=0; i<4; i++) {
      err0 += fabs(oneturnmat[4+6*i] - oneturnmat0[i]);
      oneturnmat0[i] = oneturnmat[4+6*i]; // copy
    }
  } while (deltap != 0 && err0 > 1e-15);
}
```

Laurent Deniau, CERN BE/ABP, 1211 Geneva 23, laurent.deniau@cern.ch

Emit pre-initialization Fix 1 — next release

Twiss 2

```
pro_twiss()  // ~ 400 lines
{
  …
  zero_double(orbit0, 6);
  zero_double(disp0, 6);
  zero_double(oneturnmat, 6*6);

  adjust_beam();
  probe_beam = clone_command(current_beam);
  adjust_rfc(); // init RF
  tmrefe_(oneturnmat); // ONE TURN MAP
  twcpin_(oneturnmat,disp0,r0mat,&error);
  …
  foreach deltap do {
    if (chrom_flag) {
      adjust_probe(deltap+1e-6);
      adjust_rfc();
      twiss_(oneturnmat, disp0, …); // TWISS
    }
    …
    adjust_probe(deltap);
    adjust_rfc();
    twiss_(oneturnmat, disp0, …); // TWISS
    …
  }
  …
  probe_beam = delete_command(probe_beam);
}
```

solve the fix point freq0 vs. ds

Emit 1

```
pro_emit()  // ~ 80 lines
{
  …
  zero_double(orbit0, 6);
  zero_double(disp0, 6);
  zero_double(oneturnmat, 6*6);

  adjust_beam();
  probe_beam = clone_command(current_beam);
  adjust_rfc(); // init RF
  tmrefe_(oneturnmat); // ONE TURN MAP
  twcpin_(oneturnmat,disp0,r0mat,&error);
  adjust_probe(deltap);
  adjust_rfc();
  …
  getclor_(orbit0, oneturnmat, tt, &error);
  …
  emit_(&deltap, &e_tol, orbit0, disp0, oneturnmat, …);
  …
  probe_beam = delete_command(probe_beam);
}
```

fix point

```
adjust_probe_fix_point(deltap)
{
  do { err0=0; // ds error
    tmrefe_(oneturnmat);
    twcpin_(oneturnmat,disp0,r0mat,&error);
    adjust_probe(deltap);
    adjust_rfc();
    for (int i=0; i<4; i++) {
      err0 += fabs(oneturnmat[4+6*i] - oneturnmat0[i]);
      oneturnmat0[i] = oneturnmat[4+6*i]; // copy
    }
  } while (deltap != 0 && err0 > 1e-15);
}
```

Track pre-initialization Fix 1 — next release

Twiss 2

```
pro_twiss()  // ~ 400 lines
{
  …
  zero_double(orbit0, 6);
  zero_double(disp0, 6);
  zero_double(oneturnmat, 6*6);

  adjust_beam();
  probe_beam = clone_command(current_beam);
  adjust_rfc(); // init RF
  tmrefe_(oneturnmat); // ONE TURN MAP
  twcpin_(oneturnmat,disp0,r0mat,&error);
  …
  foreach deltap do {
    if (chrom_flag) {
      adjust_probe(deltap+1e-6);
      adjust_rfc();
      twiss_(oneturnmat, disp0, …); // TWISS
    }
    …
    adjust_probe(deltap);
    adjust_rfc();
    twiss_(oneturnmat, disp0, …); // TWISS
    …
  }
  …
  probe_beam = delete_command(probe_beam);
}
```

solve the fix point freq0 vs. ds

Track 1

```
track_run()  // ~ 80 lines
{

  adjust_beam();
  probe_beam = clone_command(current_beam);
  adjust_rfc(); // init RF
  tmrefe_(oneturnmat); // ONE TURN MAP          fix point
  twcpin_(oneturnmat,disp0,r0mat,&error);
  adjust_probe(deltap);
  adjust_rfc();

  zero_double(orbit0, 6);
  zero_double(disp0, 6);
  zero_double(oneturnmat, 6*6);

  if (get_option("onepass") == 0)
    tmrefo_(&izero,orbit0,orbit,oneturnmat);
  …
  trrun_(&flag,&turns,orbit0,oneturnmat,…);
  …
  probe_beam = delete_command(probe_beam);
}
```

```
adjust_probe_fix_point(deltap)
{
  do { err0=0; // ds error
    tmrefe_(oneturnmat);
    twcpin_(oneturnmat,disp0,r0mat,&error);
    adjust_probe(deltap);
    adjust_rfc();
    for (int i=0; i<4; i++) {
      err0 += fabs(oneturnmat[4+6*i] - oneturnmat0[i]);
      oneturnmat0[i] = oneturnmat[4+6*i]; // copy
    }
  } while (deltap != 0 && err0 > 1e-15);
}
```

Laurent Deniau, CERN BE/ABP, 1211 Geneva 23, laurent.deniau@cern.ch

RF cavities, RF multipoles, Crab cavities, …

- Only RF cavities are considered for update of frequencies
  - ➡ other elements defined with **harmon** are never initialized…
- To update freq from freq0 and harmon, we should register also
  - ➡ RF multipoles
  - ➡ Crab cavities
  - ➡ Travelling Wave cavities
  - ➡ Vertical/Horizontal AC dipole
- We need to avoid nasty side effects. Quick answer says « it's safe ».

Laurent Deniau, CERN BE/ABP, 1211 Geneva 23, laurent.deniau@cern.ch