Optimizing Resonance Driving Terms Using MAD-NG Parametric Maps

L. Deniau[4], S. Kostoglou, E.H. Maclean, K. Paraschou, T. Persson and R. Tomás,
CERN, Geneva, Switzerland

Introduction

The review of the LHC octupolar resonance driving terms (RDTs) at injection was carried out in 2023[1, 2, 3, 4], motivated by the observation of undesirable losses during injection related to strongly powered octupoles, and by the will to reduce the emittance growth from e-cloud effects with weaker octupolar resonances. The MAD-NG code[5, 6] was used to simultaneously optimize the main octupolar resonances: 4Qx, 4Qy, and 2Qx-2Qy by adjusting 16 quadrupole families and 16 octupole families, for a total of 32 parameters.

To ease optimization with many knobs, MAD-NG offers a unique feature called *parametric differential algebraic maps* (parametric DA maps) build from the generalized truncated power series algebra (GTPSA) [8], which combined with other well-designed features helps simplify the overall process:

1. Load MAD-X files of LHC sequences and injection optics into `MADX` environment with appropriate setup. The circuits' logic is preserved (i.e. deferred expressions) making optimization possible.
2. Create a parametric phase-space and *link* the knobs, e.g. magnet strength circuits, to the phase-space parameters.
3. Compute the normal forms once and record the reference values of the relevant quantities to be preserved during optimization.
4. Optimize the constraints by varying the knobs using the derivatives of the relevant quantities versus the knobs, i.e. the Jacobian provided by the parametric maps.
5. Restore the knobs as scalars with optimized values, i.e. *unlink* them from phase-space parameters.

The following script snippets show the code for each of the above steps in the process.

Loading LHC Sequences & Optics (1)

```
MADX:load'lhc_seq.madx'
MADX:load'inj_optics.madx'
MADX.lhcb1.beam = beam {particle='proton', energy=450}
MADX.lhcb2.beam = beam {particle='proton', energy=450}
MADX.lhcb2.dir  = -1 -- set LHCB2 as reversed
```

Building Parametric DA Map (2)

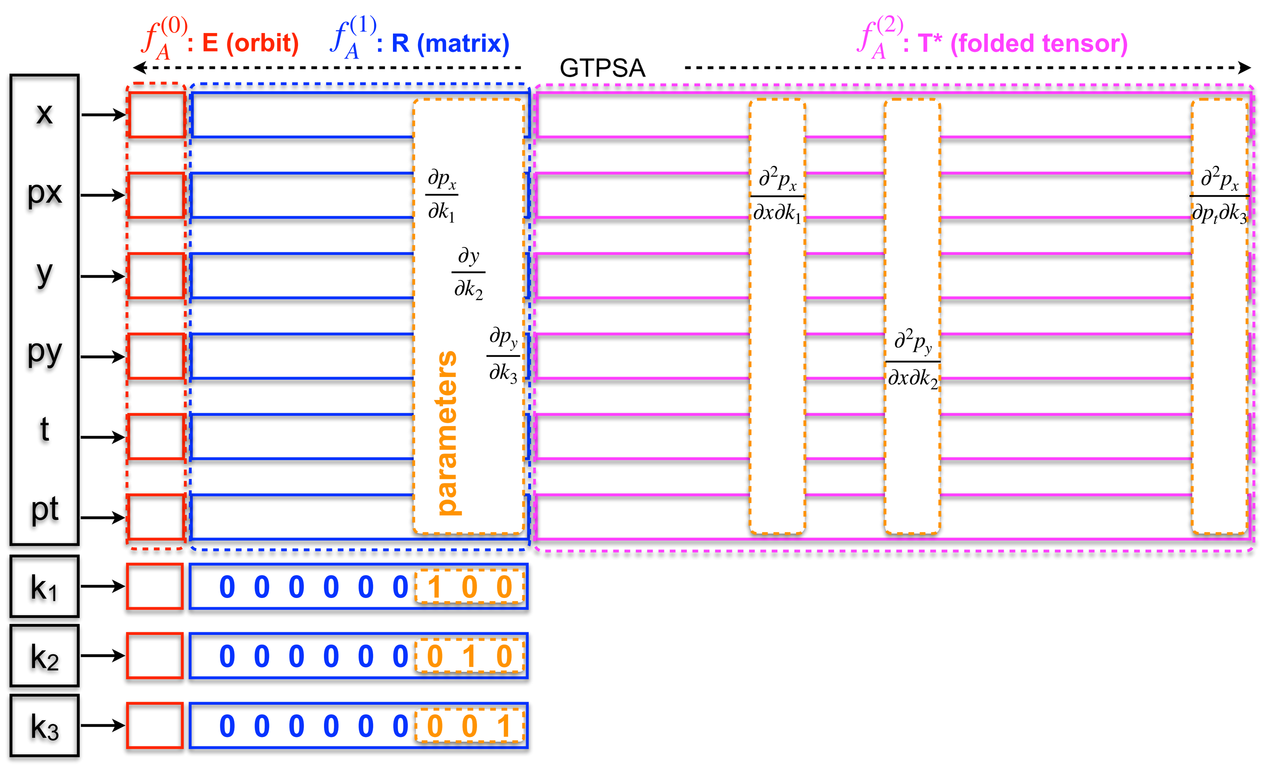
```
local prms = { -- param./knob names (strings)
  -- 16 strengths of trim quadrupoles families
  'kqtf.a12b1', 'kqtf.a23b1', ···, 'kqtf.a81b1',
  'kqtd.a12b1', 'kqtd.a23b1', ···, 'kqtd.a81b1',
  -- 16 strengths of octupoles families
  'kof.a12b1' , 'kof.a23b1' , ···, 'kof.a81b1',
  'kod.a12b1' , 'kod.a23b1' , ···, 'kod.a81b1',
}

-- DA map representing parametric phase-space
local X0 = damap {nv=6, mo=5, np=#prms, po=1, pn=prms}

-- convert scalars to GTPSAs within MADX env.
for _,knb in pairs(prms) do
  MADX[knb] = MADX[knb] + X0[knb]
end
```

Parametric Normal Forms & Setup (3)

```
-- function to compute non-linear normal forms
local function get_nf (lhc, X0)
  local _, mflw = track {sequence=lhc, X0=X0}
  return normal(mflw[1]):analyse("all")
end

-- save reference values
local nf = get_nf(X0, MADX.lhcb1)
local q1ref  = nf:q1{1}
local q2ref  = nf:q2{1}
local q1jref = nf:anhx{1,0}
local q2jref = nf:anhy{0,1}
```

Parametric DA Map Layout (GTPSA)

Schematic representation of parametric maps with 6 variables (x, p_x, y, p_y, t, p_t) of order 2 and 3 parameters/knobs (k_1, k_2, k_3) of order 1, all made from GTPSA (row-wise).



Sizes of DA maps using TPSA vs Matrix Data Structure

Number of coefficients stored in DA maps as a function of the number of variables ν and orders n using TSPA (left) and Matrix (right) representations. TPSA sizes represent an upper bound for the GTPSA.

ν \ n	1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12	13
2	6	12	20	30	42	56	72	90	110	132	156	182
3	12	30	60	105	168	252	360	495	660	858	1092	1365
4	20	60	140	280	504	840	1320	1980	2860	4004	5460	7280
5	30	105	280	630	1260	2310	3960	6435	10010	15015	21840	30940
6	42	168	504	1260	2772	5544	10296	18018	30030	48048	74256	111384
7	56	252	840	2310	5544	12012	24024	45045	80080	136136	222768	352716
8	72	360	1320	3960	10296	24024	51480	102960	194480	350064	604656	1007760

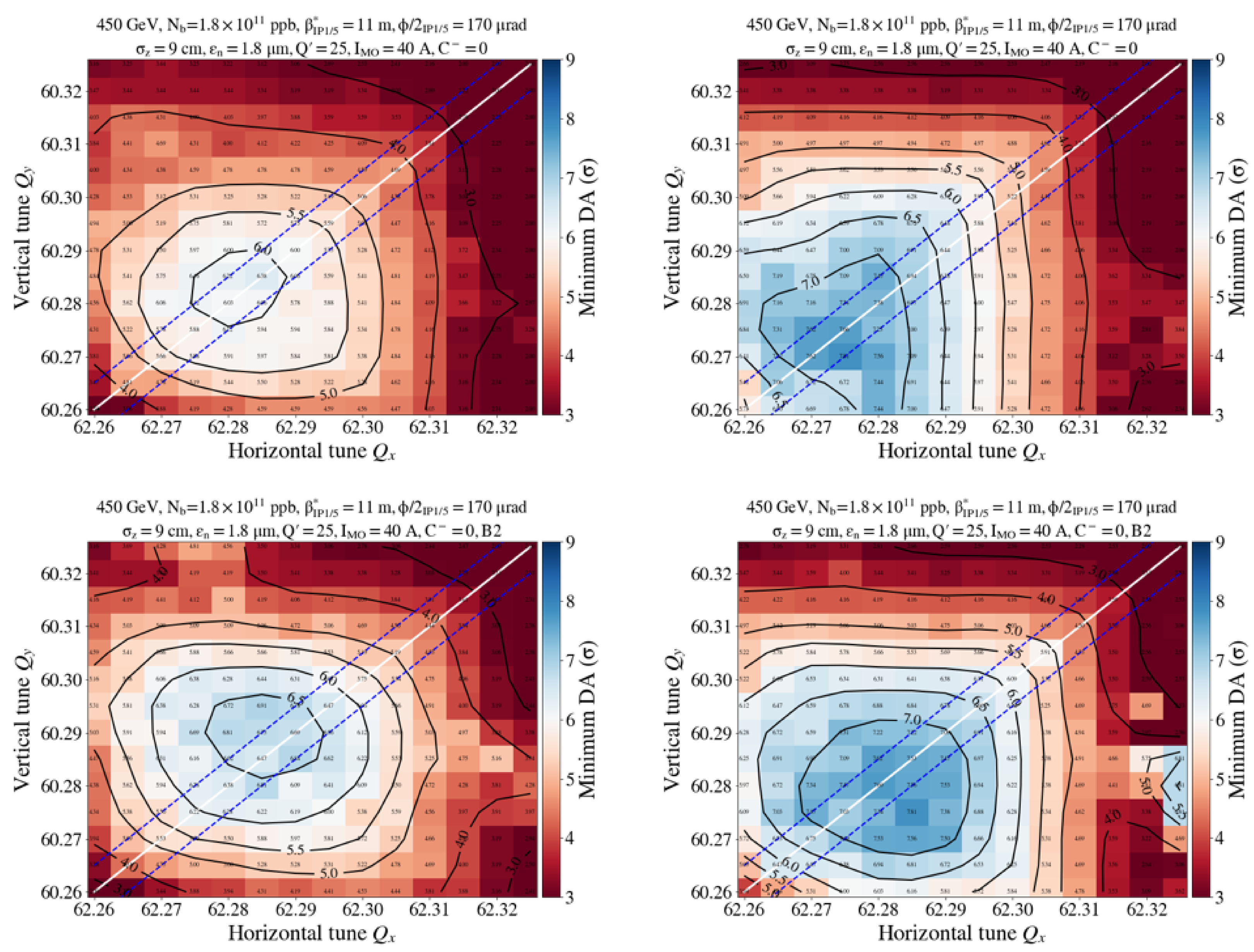
ν \ n	1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10	11
2	6	14	30	62	126	254	510	1022	2046	4094
3	12	39	120	363	1092	3279	9840	29523	88572	265719
4	20	84	340	1364	5460	21844	87380	349524	1398100	5592404
5	30	155	780	3905	19530	97655	488280	2441405	12207030	61035155
6	42	258	1554	9330	55986	335922	2015538	12093234	72559410	435356466
7	56	399	2800	19607	137256	960799	6725600	47079207	329554456	2306881199
8	72	584	4680	37448	299592	2396744	19173960	153391688	1227133512	9817068104

Optimizing RDTs (4 & 5)

```
match {
  -- compute non-linear normal forms
  command := get_nf(), -- returns nf used below

  -- compute Jacobian from parametric maps
  jacobian = \nf,_,J =>
    for k=1,32 do -- fill [10x32] J matrix
      J:set(1,k, nf:q1{1,k} or 0)
      J:set(2,k, nf:q2{1,k} or 0)
      J:set(3,k, nf:anhx{1,0,0,k})
      J:set(4,k, nf:anhy{0,1,0,k})
      J:set(5,k, nf:gnfu{"2002",k}.re)
      J:set(6,k, nf:gnfu{"2002",k}.im)
      J:set(7,k, nf:gnfu{"4000",k}.re)
      J:set(8,k, nf:gnfu{"4000",k}.im)
      J:set(9,k, nf:gnfu{"0040",k}.re)
      J:set(10,k,nf:gnfu{"0040",k}.im)
    end
  end,

  -- variables in MADX env. to use as knobs
  variables = {
    {name=prms[1] , var='MADX[prms[1]]' },
    ···,
    {name=prms[32], var='MADX[prms[32]]'},
  },

  -- target constraints as equalities to zero
  equalities = {
    {name = 'q1'    , expr = \nf -> nf:q1{1} - q1ref},
    {name = 'q2'    , expr = \nf -> nf:q2{1} - q2ref},
    {name = 'q1j1'  , expr = \nf -> nf:anhx{1,0} - q1jref},
    {name = 'q2j2'  , expr = \nf -> nf:anhy{0,1} - q2jref},
    {name = 'f2002r', expr = \nf -> nf:gnfu{"2002"}.re  - 0},
    {name = 'f2002i', expr = \nf -> nf:gnfu{"2002"}.im) - 0},
    {name = 'f4000r', expr = \nf -> nf:gnfu{"4000"}.re) - 0},
    {name = 'f4000i', expr = \nf -> nf:gnfu{"4000"}.im) - 0},
    {name = 'f0040r', expr = \nf -> nf:gnfu{"0040"}.re) - 0},
    {name = 'f0040i', expr = \nf -> nf:gnfu{"0040"}.im) - 0},
  },
} -- close match

-- restore knobs within MADX env. as scalars
for _,knb in pairs(prms) do
  MADX[knb] = MADX[knb]:get0()
end
```

Dynamic Aperture Improvements

Dynamic aperture for beam 1 (top) and beam 2 (bottom) with old (left) and new (right) injection optics for LHC. Lowering the octupolar RDTs has significantly improved the dynamic aperture at injection.



Performances

The comparison of MAD-NG versus MADX-PTC gives similar results for RDTs calculation. Reference values (3) were computed in about 0.8 s for MAD-NG and 9 s for MADX-PTC on the author's laptop, where both codes use the same physics and models, i.e. without parameters. Full optimization could also be achieved by both codes, but it took 195 s for MAD-NG using parametric maps (4), while MADX-PTC took 2730 s using finite differences, i.e. being ≈ 14 times slower overall.

References

[1] K. Paraschou, *et al.*, "Emittance from E-Cloud and Injection Optics" in LNO section meeting, 1st of February 2023. `https://indico.cern.ch/event/1248432`.

[2] L. Deniau, "MAD-NG Fast Parametric Matching to Cancel $2Q_x - 2Q_y$ and $4Q_x$ Resonances at LHC Injection", in LNO section meeting, 15th of February 2023. `https://indico.cern.ch/event/1254790`.

[3] K. Paraschou, *et al.*, "Emittance growth from electron clouds forming in the LHC arc quadrupoles", these proceedings, HB2023.

[4] R. Tomás, *et al.*, "Optics for Landau damping with minimized octupolar resonances in the LHC", these pcroceedings, HB2023.

[5] L. Deniau, "MAD-NG's Reference Manual", `https://cern.ch/mad/releases/madng/html`.

[6] L. Deniau *MAD-NG Source Repository*, `https://github.com/MethodicalAcceleratorDesign/MAD/`.

[7] E. Forest, "From Tracking Code to Analysis, Generalised Courant-Snyder Theory for any Accelerator Models", Springer, 2015.

[8] L. Deniau, C. Tomoiagă, "Generalised Truncated Power Series Algebra For Fast Particle Accelerator Transport Maps", IPAC'15, 2015. `https://cds.cern.ch/record/2141771/files/mopje039.pdf`.

[9] E. Forest, *et al.*, "Normal Form Methods for Complicated Periodic Systems using Differential Algebra and Lie Operators", Part. Accel., Vol 24, 1989. `https://indico.cern.ch/event/1180836`.