68th Adv. Beam Dyn. Workshop High-Intensity High-Brightness Hadron Beams HB2023, Geneva, Switzerland JACoW Publishing
ISBN: 978-3-95450-253-0 ISSN: 2673-5571 doi:10.18429/JACoW-HB2023-THBP15

OPTIMIZING RESONANCE DRIVING TERMS USING MAD-NG PARAMETRIC MAPS

L. Deniau*, S. Kostoglou, E.H. Maclean, K. Paraschou, T. Persson, R. Tomás
CERN, Geneva, Switzerland

Abstract

In 2023, a review of the LHC octupolar resonance driving terms at injection was carried out, motivated by two observations: (i) unwanted losses during the injection process with strongly powered octupoles and (ii) an expected reduction in emittance growth from e-cloud effects in simulations with weaker octupolar resonances. The MAD-NG code 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. These knobs were introduced as parameters in the transfer map, allowing the Jacobian required by the optimizer to be calculated in a single pass, saving 32 additional optics evaluations and avoiding finite difference approximations. Constraints on tunes, amplitude detuning, and optics around the machine were also considered as part of the optimization process. This paper reviews the parametric optimization with MAD-NG and compares the results with MADX-PTC.

INTRODUCTION

The review of the LHC octupolar resonance driving terms (RDTs) at injection was carried out in 2023 [1–4], 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 avoid any confusion, the following acronyms are used throughout this paper: MAD-X refers to the CERN code [7, 8], MADX-PTC refers to E. Forest's PTC/FPP library [9–11] embedded into MAD-X, and the keyword `MADX` refers to the special environment within MAD-NG that emulates the behavior of the global workspace of MAD-X.

To ease optimization with many knobs, MAD-NG offers a unique feature called *parametric differential maps* build from the generalized truncated power series algebra (GTPSA) [12], 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.
2. Create a parametric phase-space and *link* the knobs, e.g., magnet strengths, to the phase-space parameters.
3. Optimize the constraints by varying the knobs using the derivatives of relevant quantities versus these knobs.
4. Restore the knobs as scalars with optimized values.

Each of these steps will be detailed in the following sections, starting from the preliminary setup performed by the script snippet below, where the `LHCB1` and `LHCB2` sequences and the optics file of MAD-X are directly read, translated and loaded into the special `MADX` environment in MAD-NG. The sequences are modeled with thick lens elements and a proton injection beam is created and attached to them:

```
local pb450=beam{particle='proton',energy=450}
MADX:load'lhc_seq.madx'
MADX:load'inj_optics.madx'
MADX.lhcb1.beam = pb450
MADX.lhcb2.beam = pb450
MADX.lhcb2.dir  = -1 -- set LHCB2 as reversed
```

PARAMETRIC MAPS

Setting up parametric phase-space in MAD-NG is simple, even when everything has been loaded inside the `MADX` environment. For convenience, we start by defining the lists of variable and parameter names of the 6D parametric phase-space such that they can be accessed by name everywhere:

```
local vars = { -- variable names (strings)
  -- 6 canonical variables of phase-space
  'x','px','y','py','t','pt'
}
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',
}
```

From these lists, we can define the parametric phase-space with `nv=6` variables of order `mo=5` and `np=32` parameters of order `po=1` named after the knobs in the optics file, and where the list of names must satisfy `#vn=nv+np`. The order of the *variables* must be `mo=4+1` as we want to optimize octupolar resonances at order 4 using their derivatives versus the knobs from order 5.

```
-- DA map representing parametric phase-space
local X0 = damap {nv=#vars,np=#prms,mo=5,po=1,
                  vn=tblcat(vars,prms)}
```

The next step is to link the knobs defined in the optics file to the parameters defined in the phase-space by replacing the knobs (scalars) in the `MADX` environment with their corresponding parameters (GTPSA) from the phase-space. Thanks to MAD-NG's deferred expressions and physics support for polymorphism, which simplifies the whole process through automatic handling by the `track` and `twiss` commands. This is achieved by the loop hereafter that replaces the knobs in `MADX` by the sum of knobs and parameters with

* laurent.deniau@cern.ch

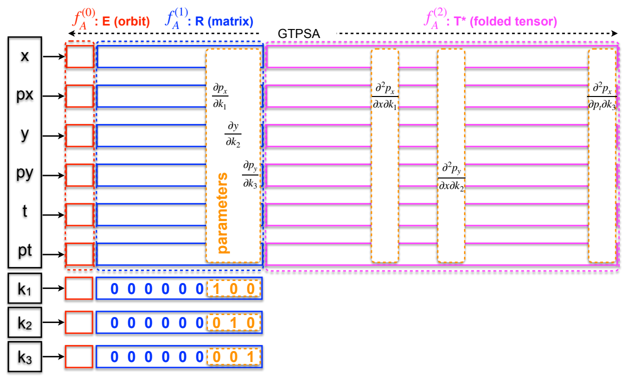

$f_A^{(0)}$: E (orbit)
$f_A^{(1)}$: R (matrix)
GTPSA
$f_A^{(2)}$: T* (folded tensor)
x
px
y
py
t
pt
k1
k2
k3
$\frac{\partial p_x}{\partial k_1}$
$\frac{\partial y}{\partial k_2}$
$\frac{\partial p_y}{\partial k_3}$
parameters
$\frac{\partial^2 p_x}{\partial x \partial k_1}$
$\frac{\partial^2 p_y}{\partial x \partial k_2}$
$\frac{\partial^2 p_x}{\partial p_t \partial k_3}$
0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 1

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

identical names such that initial conditions (e.g., injection optics) are preserved. It is important to note that any attribute of the lattice elements that might affect the constraints can be used transparently as parameters, not only the strengths.[1]

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

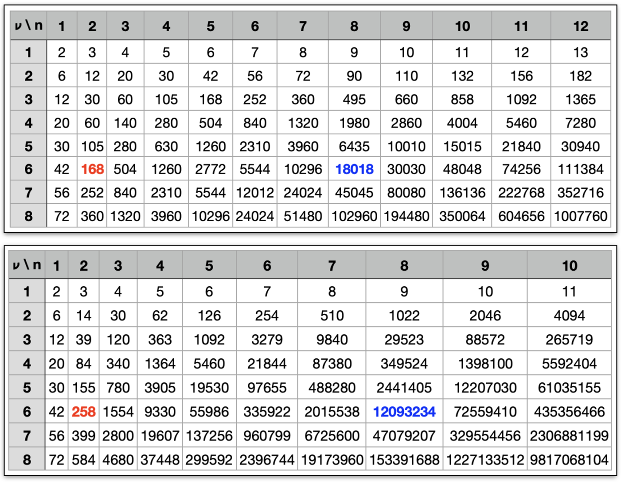
Figure 1 shows a schematic representation of a phase-space of 6 variables of order 2 and 3 parameters of order 1. Each *variable* is a row-wise GTPSA that represents a truncated analytic function in the variables and parameters as multivariate Taylor polynomials. Each *parameter* is defined as GTPSA of order 1, independently of the order of the parameters in the variables set by `po`, with their first derivative set according to its position in the phase-space. Matrix codes like MAD-X use similar representation but are organized block-wise as shown by the orbit column vector E, the 1st derivatives matrix R, and the 2nd derivatives tensor T. This representation is not well suited for higher orders as shown in Fig. 2 where tensor sizes grow exponentially, making it difficult to carry out optics calculation at order 4. But MADX-PTC also uses TPSA to perform such high order computation, but supporting hundreds of variables and parameters with non-uniform orders is a unique feature of the GTPSA library [6, 12] which is an integral part of MAD-NG.

PARAMETRIC NORMAL FORMS

In order to optimize the octupolar resonances, we need to *analyze* the non-linear *normal* forms of the parametric one-turn map obtained from the `track` command, and performed by the function `get_nf` below. Since the objectives imply keeping the tunes and the amplitude detuning under control, we need to run `get_nf` once to save these reference quantities.

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

[1] Lengths and positions might also be useful parameters for optimization.



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

Figure 2: Number of coefficients stored in DA maps as a function of the number of variables ν and orders n using TSPA (top) and Matrix (bottom) representations.

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

The following MADX-PTC macro is equivalent to the `get_nf` function for comparing the quantities obtained:

```
get_nf(seqname) : macro = {
  use, sequence=seqname;
  ptc_create_universe;
  ptc_create_layout, model=2, method=4,
                     exact=true, time=true;
  select_ptc_normal, q1=0, q2=0;
  select_ptc_normal, anhx=1,0,0;
  select_ptc_normal, anhy=0,1,0;
  select_ptc_normal, gnfu=4,0,0;
  select_ptc_normal, gnfu=-4,0,0;
  ptc_normal, normal, no=4, icase=56;
  ptc_end;
}
```

The algorithm for the calculation of the non-linear normal forms can be found in Refs. [11, 13]. MAD-NG's parametric non-linear normal forms and analysis implementations are variants based on the Lie operators of the GTPSA library [12], and widely presented [2, 14–18].

PARAMETRIC OPTIMISATION

The main difference between the optimization with MAD-NG and MADX-PTC is the order of the DA map representing the phase-space, which is one order higher in the former to extract the *exact* Jacobian from the fifth order of the parametric normal form. The downside is that for each iteration of the optimizer, MAD-X has to run the MADX-PTC script above 1+32 times to *approximate* the Jacobian with finite differences, whereas MAD-NG only needs to run it once. If we consider the map sizes ratio between order 4 with no parameter and order 5 with 32 parameters to be $1260 \times 33/(2772 + 1260 \times 32) \approx 0.97$, and a computational time almost linear in size, the two methods should be equivalent. However, while MAD-NG physics is the same as MADX-PTC with `exact` and `time` set to `true` as in the `get_nf` macro above, it is about 30 times faster on average.

The following script shows the `match` command[2] being executed in an attempt to satisfy constraints specified as *equalities* and expressed as lambda functions returning values to be compared with zero, i.e., the targeted/reference values must be subtracted. The optimizer will vary the values of the *variables* according to results returned by the *command* invoked once (or 33 times) for each iteration.

```
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},
  {name='f2002i',expr=\nf->nf:gnfu"2002".im)},
  {name='f4000r',expr=\nf->nf:gnfu"4000".re)},
  {name='f4000i',expr=\nf->nf:gnfu"4000".im)},
  {name='f0040r',expr=\nf->nf:gnfu"0040".re)},
  {name='f0040i',expr=\nf->nf:gnfu"0040".im)},
},
} -- close match
```

The key point is the function `jacobian` which is called once (if present) in place of the 33 calls to obtain the Jacobian `J` required by the optimizer. It queries the analyzed normal forms with the extra argument `k` representing the index of the parameter, e.g., `nf:gnfu{"2002",k}` returns the complex value $\partial f_{2002}/\partial K_k$ from fifth order, where K_k is the strength associated with the k-th parameter of the phase-space.

Finally, once a suitable solution has been found, we restore the knobs/strengths as scalars in the `MADX` environment for further use.

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

[2] MAD-NG's match command follows the same "structure" as in MAD-X.

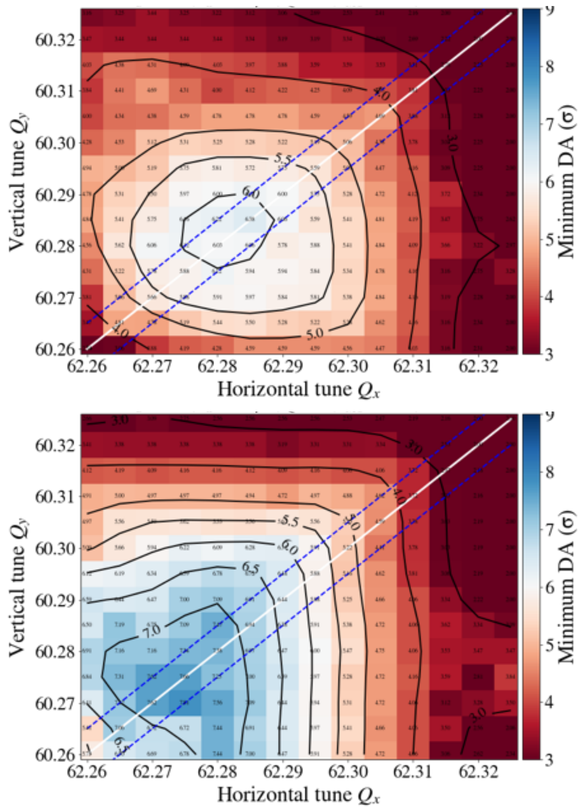


Figure 3: Dynamic aperture for beam 1 old (top) and new (bottom) LHC injection optics (beam 2 has similar results).

COMPARISON WITH MADX-PTC

The comparison of MAD-NG versus MADX-PTC gives similar results for RDTs calculation. The reference values are computed in about 9 s on the author's laptop for both codes, however, MAD-NG tracks a fifth-order phase-space with 32 parameters which involves about 34 times as many calculations. Full optimization could also be achieved by both codes, but it took 21 evaluations of `get_nf` and 195 s for MAD-NG, while MADX-PTC took 342 evaluations and 2730 s to converge, i.e., being ≈ 14 times slower overall.

The key figure of merit of an optics design in the non-linear regime is the Dynamic Aperture (DA), i.e., the minimum transverse amplitude at which particles start to become unstable. Figure 3 shows the DA for the old (top) and the new (bottom) optics versus horizontal and vertical tunes, demonstrating the superiority of the new optics with optimized octupolar resonances. The operational experience with the new injection optics is reported in Ref. [4], demonstrating an improvement on lifetime.

CONCLUSION

We detailed the procedure for optimizing the main octupolar resonances 4Qx, 4Qy, and 2Qx-2Qy for the LHC injection optics, taking advantage of MAD-NG parametric maps and normal forms. We also showed how this optimization improved the dynamic aperture for beam 1 with similar results for beam 2. Further work will be done soon on the optimizer of MAD-NG and compared to MAD-X.

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", presented at HB23, Geneva, Switzerland, Oct. 2023, paper THBP16, these proceedings.

[4] R. Tomás *et al.*, "Optics for Landau damping with minimized octupolar resonances in the LHC", presented at HB23, Geneva, Switzerland, Oct. 2023, paper THBP20, these proceedings.

[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] MAD-X Home Page, http://cern.ch/madx

[8] MAD-X Source Repository, https://github.com/MethodicalAcceleratorDesign/MAD-X/

[9] PTC/FPP Source Repository, https://github.com/jceepf/fpp_book

[10] E. Forest *et al.*, "Introduction to the Polymorphic Tracking Code: Fibre Bundles, Polymorphic Taylor Types and Exact Tracking", CERN-SL-2002-044-AP, KEK-REPORT-2002-3, 2002. https://cds.cern.ch/record/573082

[11] E. Forest, *From Tracking Code to Analysis, Generalised Courant-Snyder Theory for any Accelerator Models*, Springer, 2015. doi:10.1007/978-4-431-55803-3

[12] L. Deniau and C. I. Tomoiaga, "Generalised Truncated Power Series Algebra for Fast Particle Accelerator Transport Maps", in *Proc. IPAC'15*, Richmond, VA, USA, May 2015, pp. 374–377. doi:10.18429/JACoW-IPAC2015-MOPJE039

[13] E. Forest *et al.*, "Normal Form Methods for Complicated Periodic Systems using Differential Algebra and Lie Operators", *Part. Accel.*, vol. 24, pp. 91–107, 1989.

[14] L. Deniau, "Normal Forms in MAD-NG", in LNO section meeting, 27th of May 2021. https://indico.cern.ch/event/1041358

[15] L. Deniau, "Normal Forms Implementation in MAD-NG", in LNO section meeting, 15th of September 2021. https://indico.cern.ch/event/1076583

[16] L. Deniau, "MAD-NG Normal Forms, RDTs and Amplitude Detuning vs MADX-PTC", in LNO section meeting, 10th of December 2021. https://indico.cern.ch/event/1104547

[17] L. Deniau, "Update on Normal Forms in MAD-NG, Normal Forms Analysis and the Log of Lie Maps", in LNO section meeting, 8th of April 2022. https://indico.cern.ch/event/1147560

[18] L. Deniau, "Update on Normal Forms in MAD-NG, Parametric Normal Forms and RDTs applied to FCC-ee", in LNO section meeting, 13th of July 2022. https://indico.cern.ch/event/1180836