Beam Physics Note
TRI-BN-25-11
May 30, 2025

Sequential Optimization from IOS to MEBT

Omar Hassan, Olivier Shelbaya, Thomas Planche

TRIUMF

Abstract: Building on MCAT's sequential optimization capability, based on `Python` scripting, this technique is now included natively within `TRANSOPTR`. Sequential optimization tackles multiple tuning tasks in a single pass leading to more performant online optimization. For the IOS-MEBT section at ISAC, it reproduces Baartman's design tune in a fraction of the time of the MCAT approach.

4004 Wesbrook Mall, Vancouver, B.C. Canada V6T 2A3 · Tel: 604 222-1047 · www.triumf.ca

1 Introduction

The beam envelope code TRANSOPTR [1] was recently upgraded to natively support sequential optimization [2]. Previously, TRANSOPTR could only handle one optimization at a time, now up to 9 problems can be done in sequence, with the method implemented directly into TRANSOPTR's FORTRAN source code.. In this report, an application of this sequential optimization upgrade will be shown where the sections from IOS to the first MEBT section are optimized, there is a discussion of the sequences chosen and the elements used for them, and finally, profiling is done to carry out a comparison with the pythonic approach currently found in MCAT [3].

2 Sequential Optimization

Sequential optimization allows for the solution of several different tuning problems in a charged particle accelerator and transport system. This is achieved with a single execution of the envelope code, which progressively solves each sub-problem and moves on to the next, while retaining the optimum values found at each step. The idea was first used in MCAT through a Python script and is now built directly into TRANSOPTR. Because each sequence can target a different portion of the beamline or pursue a different objective, the lattice is adjusted online and convergence on the user-defined constraints occurs quickly. The rest of this section explains how to use sequential optimization, with examples shown from the IOS to MEBT beamline. To use seqopt, include this line in the COMMON blocks of the sy.f file: `COMMON/CSEQOPT/I_SEQ`

2.1 Usage in data.dat

Column 4 acts as a "sequence selector": it tells the optimizer which element should be used for each tuning sequence. In earlier versions of TRANSOPTR (see § 3.2.9 of the manual) this column was a Boolean flag, 0 or 1, indicating whether the element was tunable for fitting. It has since been upgraded to accept an integer code so to account for multiple optimization problems. Because the code is interpreted digit-by-digit rather than as a single number, a value such as 34 will allow the element to be tuned in both sequence 3 and sequence 4, where the digits are treated as a set, not as the integer thirty-four. An important point to note is that only 9 sequences are currently supported. An example can be found in figure 1.

```
0.0     0.0     5000.0    2 ! IOS:Q9:POS:VOL QE1 V
0.0     0.0     10000.0   0 ! IOS:B10:POS:VOL QX0 V
0.0     0.0     10000.0   0 ! IOS:B10:NEG:VOL QX1 V
0.0     0.0     10000.0   1 ! IOS:Q10:POS:VOL QE2 V
0.0     0.0     5000.0    1 ! IOS:Q11:POS:VOL QE3 V
0.0     0.0     10000.0   0 ! IOS:B13:POS:VOL QX2 V
0.0     0.0     5000.0    2 ! ILT:Q33:POS:VOL QE4 V
0.0     0.0     5000.0    3 ! ILT:Q34:POS:VOL QE5 V
0.0     0.0     10000.0   34 ! ILT:Q35:POS:VOL QE6 V
0.0     0.0     10000.0   34 ! ILT:Q36:POS:VOL QE7 V
0.0     0.0     10000.0   34 ! ILT:Q37:POS:VOL QE8 V
```

Figure 1: data.dat snippet showcasing the use of native sequential optimization.

2.2 Subroutines

The following subroutines have been added to TRANSOPTR's source code: `fit_seq`, `fitarb_seq`, `twissmatch_seq`, `waist_seq`. The `I_SEQ` variable stores the identifier of the sequence presently being optimised which allows for the use of conditional statements to contain any fitting command presently available in TRANSOPTR, or others such as prints or return statements. See figure 2 for an example.

```
! sequential optimization fits
call fit_seq(2,1,2,1,0.,1.,1)
call fit_seq(2,1,4,3,0.,1.,1)
call fit_seq(1,2,2,6,0.0,1.,1)
call drift(0.09996,".")
call drift(8.3609,".")
call fit_seq(1,2,1,6,0.0,1.,1)
! endOf_ios_db10
! fringeQ
call fringeQ(0.085, 0.004, 0.031, -0.232)
IF(I_SEQ.EQ.1) return   ! conditional with I_SEQ
```

Figure 2: sy.f snippet with sequential fits and usage of `I_SEQ` shown.

3 Results

To test the new native sequential optimization upgrade in TRANSOPTR, the lattice stretch from the Ion Source (IOS) to the Medium-Energy Beam Transport (MEBT) section was chosen. This is a relatively complex lattice including many fitting objectives, which takes roughly 10 seconds to tune using MCAT's approach. There are also many sequences which makes this a good test for this upgrade.

Table 1 shows the full breakdown of all the sequences with the optimization goals, alongside the elements used in each sequence. This is also shown visually in figure 3. Figure 4 shows a comparison with the design tune by Rick Baartman, sequential optimization closely matches this tune with a few notable differences, including: 1) Less symmetric achromats but much smaller M_{16} component for the first achromat. 2) Slightly weaker periodicity before the IRA section. 3) Different envelopes at MEBT as a result of a different match into the RFQ.

An approximate time estimate shows that the native TRANSOPTR approach completes the optimization to the RFQ match in a fraction of a second, and the full optimization takes about 2.5 seconds. The native implementation cuts runtime by roughly a factor of four, giving the MCAT control-room application a performance boost.

Table 1: Sequential–optimization steps from IOS to MEBT

Seq.	Quadrupoles in fit	Goal
1	IOS:Q10–Q13	First achromat quads; force $\partial_s M_{16}$ and M_{16} to 0.
2	IOS:Q9, IOS:Q33	Quads before and after achromat; zero the transverse correlations R_{12} and R_{34}.
3	ILT:Q34–Q41	Triplet plus end quads; create a $\beta = 50$ cm waist, matching optics from Q34 to Q41.
4	ILT:Q35–Q40	Triplet only; refine the $\beta = 50$ cm waist.
5	ILT:Q43–Q46	Second achromat quads; again drive $\partial_s M_{16}$ and M_{16} to 0.
6	ILT:Q42, ILT:Q47	Quads before and after achromat; zero the transverse correlations R_{12} and R_{34}.
7	ILT:Q47–Q50	Periodic section; Twiss-match Q47 $\rightarrow$ Q42, Q48 $\rightarrow$ Q41, Q50 $\rightarrow$ Q48.
8	IRA:Q1–Q4	Match into the RFQ.
9	MEBT:Q1–Q5	Match into the MEBT.

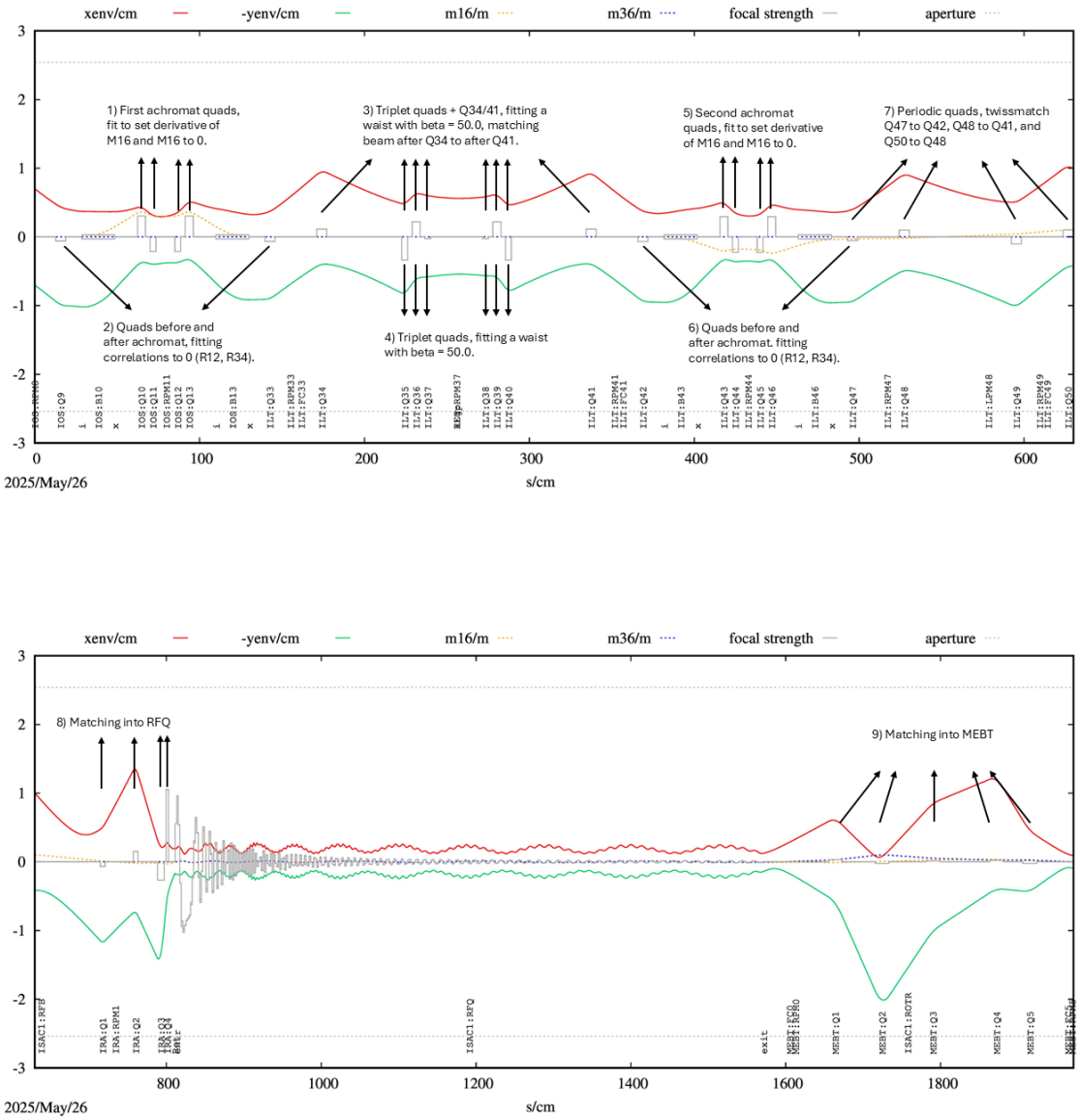

Figure 3: Breakdown of sequences 1-9 using sequential optimization.

3.1 Comparison with Design Tune

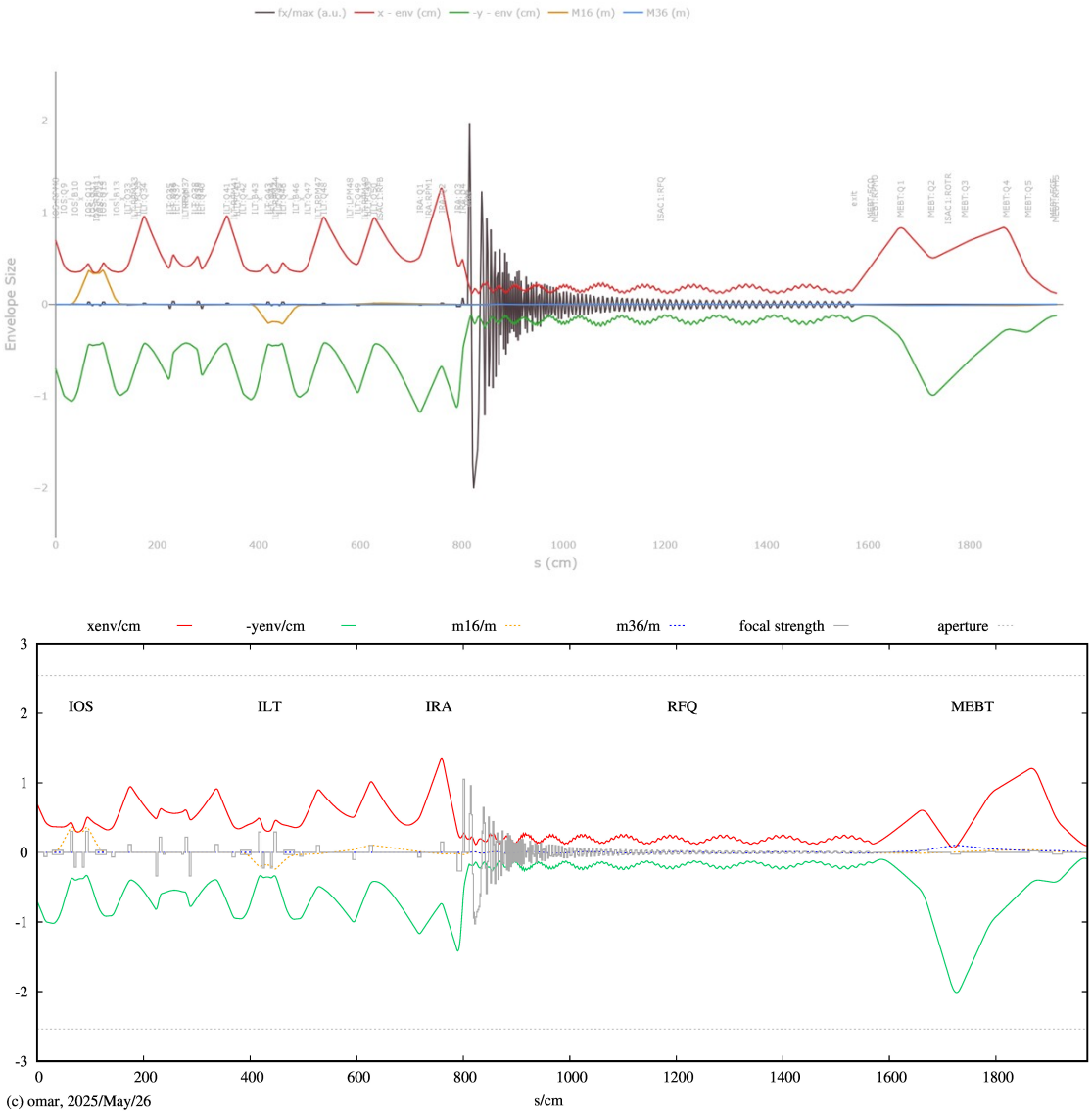

Figure 4: Full tune from IOS to MEBT. **Top:** Rick Baartman's Design Tune. **Bottom:** Sequential optimization tune, focal strengths for the RFQ are scaled down to increase readability.

3.2 Code

The TRANSOPTR files for this work can be found here: https://gitlab.triumf.ca/omar-archive/seqopt/-/tree/main/ios_mebt?ref_type=heads.

References

[1] E. A. Heighway and M. S. de Jong. Transoptr: A Beam Transport Design Code with Space Charge, Automatic Internal Optimization and General Constraints. *Technical Report, Atomic Energy of Canada, Limited*, 6 1984.

[2] Olivier Shelbaya. Sequential Tune Optimization with TRANSOPTR. Technical Report TRI-BN-20-14, TRIUMF, 2020.

[3] Olivier Shelbaya. Model coupled accelerator tuning (phd dissertation). Technical Report PhD Dissertation, TRIUMF, UVic Dept. of Physics & Astronomy, 2023.