Beam Physics Note
TRI-BN-20-14
November 5, 2020

Sequential Tune Optimization with TRANSOPTR

Olivier Shelbaya

TRIUMF

Abstract: Using an XML repository of accelerator and beamline device locations and properties, which now includes location-specific constraints upon the sigma and transfer matrices, the envelope code TRANSOPTR can be used to perform long, multi-staged tune optimizations. This opens the door to the development of an on-line tune re-computation capability, allowing the easy generation of accelerator tunes, at different charge-to-mass ratios, tailored to specified σ-matrix parameters.

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

1 Introduction

In the course of developing the modelling capability for the ISAC-I accelerator, I wanted to implement a tool which could aid in the computation of start-to-end tunes for the machine. As the reader may know, the ISAC-I linac comprises several systems and components, which are somewhat laborious to expose at the introduction of each report. Summaries are available in several TRIUMF beam physics reports, but an overview is provided in [1].

Each section's unique constraints upon the σ-matrix can be automatically added to `TRANSOPTR` through inclusion in our `XML`-formatted element database, which tabulates the location, optical parameters and control system variables of each device comprising the TRIUMF accelerators and their beamlines. From this, the `python` package `xml2optr`[1] automatically generates `TRANSOPTR` files necessary to simulate a section of the system, whose start and end locations are user-specified. Parameters representing the beam distribution are also user supplied, as is an initial tune, or collection of device setpoints specified to `TRANSOPTR`.

As ISAC is a multi user, multi beam, multi A/q facility, the finding and saving of accelerator and beamline tunes is a critical activity which is also frequently carried out. The existing strategy consists of scaling saved reference tunes, which have previously been carefully established on-line with a specified beam at a specified energy. When it is time to deliver another mass-to-charge beam, the reference tune's fields are scaled to produce the equivalent effect upon the beam.

Multi-A/q delivery also entails constant changes in ion source configuration, which may result in a considerable alteration of source operating parameters. Within OLIS, this may also mean switching between any of the surface, microwave or multi-charge ion sources contained within the OLIS structure. On-line, this results in a necessity for operators to make use of so-called matching sections, designated groups of quadrupoles reserved for manual beam matching into the scaled reference transport tune. At OLIS, these have generally consisted of quadrupoles IOS:Q1 up to IOS:Q8.

Operators will freely admit that the necessity for manual quadrupole tuning in such sections is constant - and considerable time is devoted to it. As the bulk of pilot beam tuning is carried out on a mix of transmission and beam profile monitoring, the necessity of constantly tuning quadrupoles evidences a need to alter the overall transfer matrix of the system. This, in and of itself, is evidence that the initial σ-matrix elements change from one source configuration to the next. Depending on the precise source configuration, such a mismatch may not be trivial to overcome manually, with available diagnostics.

Such an operational reality calls for the capability to constantly re-compute tunes, assuming a reliable method to extract the beam distribution on-line. This report will present such a capability, in which parameters representing the tune constraints are added directly to the accelerator database. In this manner, the generation of `TRANSOPTR` files automatically includes those tune constraints, properly positioned within the element lattice.

Using supplementaly developed `python` software, `TRANSOPTR` is sequentially executed with groups of constraints activated and groups of devices set to be optimized, at each step. The optimized device setpoints are fed back into `TRANSOPTR`, which moves onto the next block to optimize. In a manner, the code crawls along the accelerator, computes the local tune and moves onto the next section.

[1] Credit: Paul Jung TRIUMF/UWaterloo

Devices are set to uniform starting values, which removes the necessity of storing pre-existing best-guess tune values. This method pre-assumes no existing device setpoints for the sections. In this manner, TRANSOPTR can be called upon to re-compute the tune of the ISAC-I accelerator, independent of charge-to-mass.

2 Model Coupled Accelerator Tuning Constraints

The python subroutine dubbed MCATSequencer, whose operating principle is reported here, is in ongoing development. This should be thought of as a test implementation, useful for an initial demonstration of model coupled accelerator tuning.

In the acc/ database, in which element descriptions are stored, the constraints are coded in directly. An example in the ISAC-MEBT section is shown below. This particular example is located inside sequence mebt_db0.xml, previously reported in [2] and corresponds to a round, single achromatic focus at the mid-point of the MEBT rebuncher cavity.

```
<element id="vat" type="fortline" s="7220.0972*mm" l="0">
<optr>! MCAT-constraint: 0,1,0,0,0,0,0,0,0,0,0,0,0,0</optr>
<optr>!nlines: 5</optr>
<optr>!call dr(9.96,'half-rebuncher')</optr>
<optr>!call twissmatch(1, 0.0, bfoc, 1., 1)</optr>
<optr>!call twissmatch(3, 0.0, bfoc, 1., 1)</optr>
<optr>!call fit(1,1,6,0.0,1.,1)</optr>
<optr>!  return</optr>
<notes>Olivier: Nov 4, 2020: MEBT round (x,y) match, single achromatic
       at MEBT rebuncher cavity.</notes>
</element>
```

Note that TRANSOPTR is run in FORTRAN, though fully wrapped in python. However, the lines that define the constraint, nested between <optr> XML tags, are commented out with the ! character. Generation of simulation files always includes the constraints but leaves them commented out by default, rendering them transparent to manual TRANSOPTR use.

The subroutine MCATSequencer simply parses through the file sy.f, locates blocs of tune optimization constraints, thanks to their standard header which is read and interpreted. In the above example, the full tuning sequence consists of 14 levels of operations, or separate steps. This is encoded in the first line included to TRANSOPTR, with each character following the semicolon being a boolean, representing whether or not this particular block of constraints gets uncommented at each operation.

Within MCATSequencer, each level of operation is represented by a dictionary whose keys are the number of each step. The associated dictionary entry is a list of TRANSOPTR variables which are set to be optimized during that particular step. It is presently incumbent upon the user to ensure proper sequential definition of variables for each step. In other words, there is no automatic verification that the specified variables are actually included within the section of devices to be optimized.

Using a `redis` database in parallel, a virtual control system of sorts is created. Within the database, each process variable's corresponding EPICS name is stored and to each is associated a setpoint. At the end of each sequential tune optimization, `MCATSequencer` updates the `redis` database values for each PV. In other words, it is now trivial to interface with the actual control system for the purpose of on-line development, should this database's buffer then be loaded to EPICS using existing software at TRIUMF [3].

3 Conclusion: Two Examples at ISAC

Here, `MCATSequencer` has been run for both the low energy injection tune into the ISAC-RFQ, shown in Figure 1 and for the medium to high energy section, spanning MEBT, DTL and HEBT, shown in Figure 2. In the latter, the DTL cavities are left off, though the `TRANSOPTR` model is capable of simulating full acceleration [4].

In all cases, beams of a diversity of A/q's and energies are simulated, along with varied starting beam distributions. The sequential optimization procedure was carried out, allowing for the generation of all datasets shown in both Figures 1 and 2 in a matter of minutes. This is intended to play a central role in the implementation of model coupled accelerator tuning on-line at ISAC.

References

[1] Olivier Shelbaya. ISAC-I RF Acceleration. Technical Report TRI-BN-18-02, TRIUMF, 2018.

[2] Olivier Shelbaya. TRANSOPTR Implementation of the MEBT Beamline. Technical Report TRI-BN-19-02, TRIUMF, 2019.

[3] Barquest C. *Web-Based Control Room Applications at TRIUMF*. In *Proceedings of the 9th International Particle Accelerator Conference*, pages 4832–35, 2018.

[4] Olivier Shelbaya. The TRANSOPTR Model of the ISAC Drift Tube Linear Accelerator - Part I: Longitudinal Verification. Technical Report TRI-BN-20-08, TRIUMF, 2020.

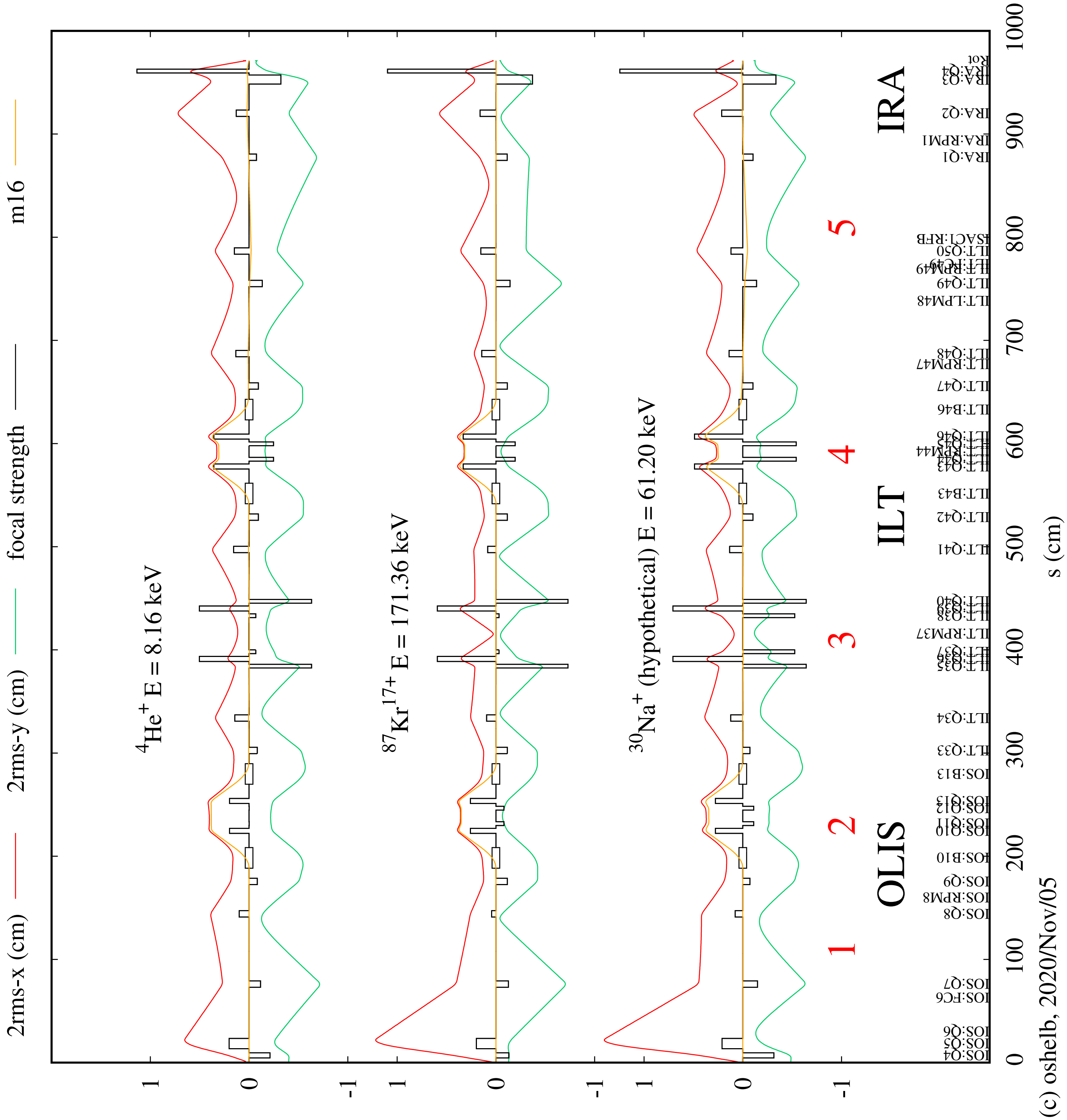

Figure 1: `MCATSequencer` generated `TRANSOPTR` 2rms beam envelope simulations from OLIS collimator 3B (downstream of the OLIS dipole) for three specified beams, for different charge-to-mass ratios and energies, each corresponding to E/A = 2.04 keV/u, for ISAC-RFQ injection. In each case, the particular tune, or collection of element values has been found by `TRANSOPTR`'s internal optimizer. Each computation was started with uniform quadrupole values. Tune constraints generated through `acc/` database. The sequence of tuning parameters is roughly outlined in red, from 1 to 5.

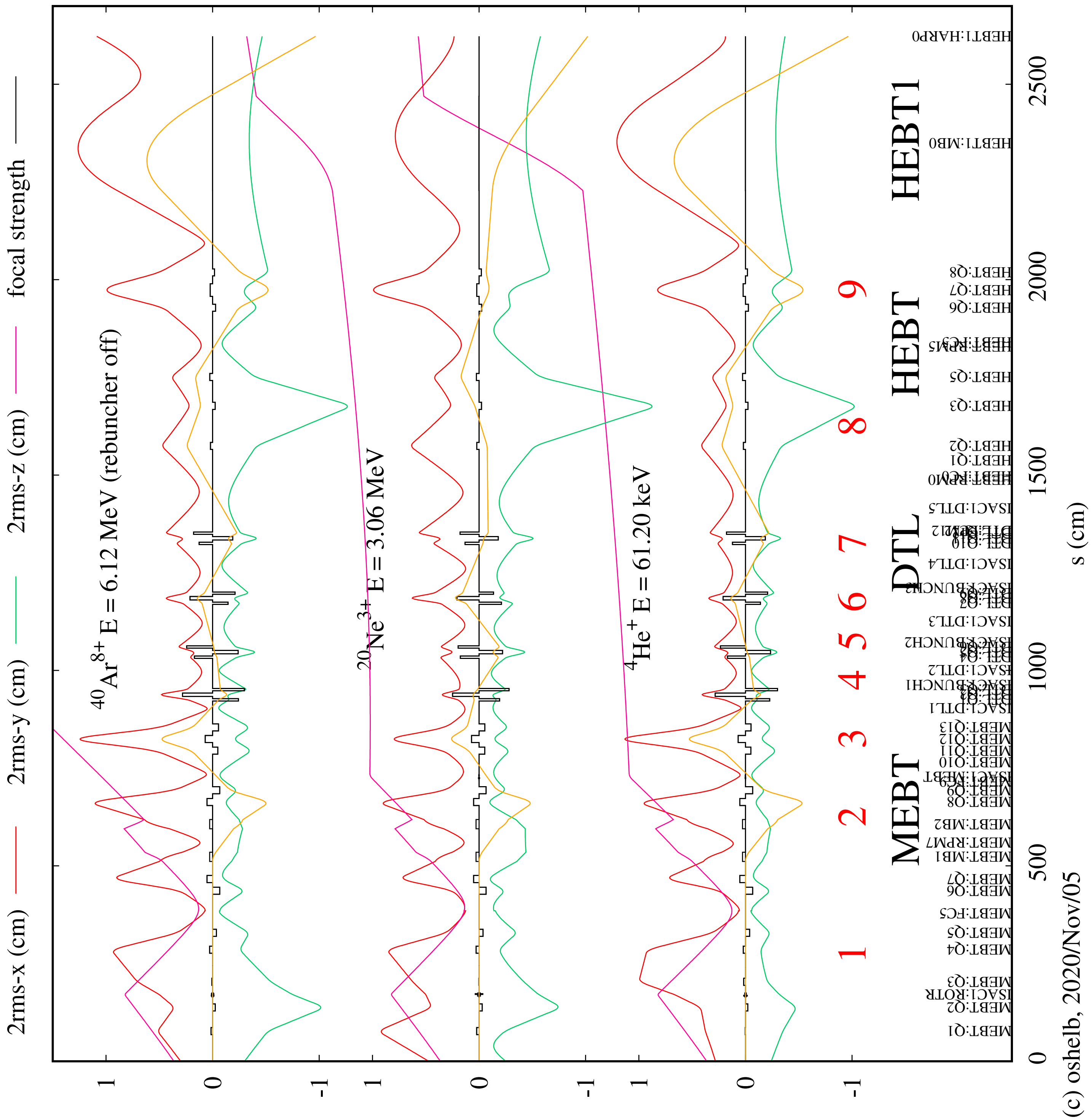

Figure 2: `MCATSequencer` generated `TRANSOPTR` 2rms beam envelope simulations from RFQ output to HEBT1 high energy diagnostic station (Prague magnet). Three specified beams are shown, for different charge-to-mass ratios and energies, each corresponding to E/A = 0.153 MeV/u, ISAC-RFQ output energy. In one scenario, the rebuncher was left off. The DTL RF cavities are unpowered in this example. In each case, the particular tune, or collection of element values has been found by `TRANSOPTR`'s internal optimizer. Each computation was started with uniform quadrupole values. Tune constraints generated through `acc/` database. The sequence of tuning parameters is roughly outlined in red, from 1 to 9 (DTL RF steps omitted).