Beam Physics Note
TRI-BN-22-06
March 8, 2022

Generation of TRANSOPTR files with xml2optr

Olivier Shelbaya, Paul M. Jung

TRIUMF

Abstract: This short note is intended as a use example for the python package xml2optr, which enables conversion of TRIUMF-acc/ repository XML files into standard TRANSOPTR data.dat and sy.f files for envelope simulations and tune computations. In this particular example, a tunefile containing beam and tune information is used.

Prerequisites

- An account on `gitlab.triumf.ca`, requiring a TRIUMF-TRIDENT ID[1]
- A unix-compatible operating system (Linux, MacOS)[2]
- `python3` and `pip3` installed on your machine
- gitlab-ssh must be configured on your system, instructions here
- git-cloned copy of `acc` available here
- git-cloned copy of `accpy` available here
- git-cloned copy of `xml2optr` available here
- git-cloned copy of `TRANSOPTR` available here, with basic use instructions listed in [1]

Note that each of the above git repositories have build-dependencies that you must install. For simplicity, it is strongly recommended that you crate a directory for all of your high level application software:

```
mkdir ~/hla/
```

Place all your git repositories in that folder. If you're unsure which shell you're using, use the following command in a terminal:

```
echo $SHELL
```

Together with this, you should define environment variables:

```
export MYAPPSDIR='/path/to/hla/'
export ACCDIR='/path/to/acc/'
```

Create an alias to run `xml2optr` from your terminal:

```
alias xml2optr='python3 $MYAPPSDIR/xml2optr/bin/run-xml2optr'
```

Don't forget to source your definition file (in `.bashrc` or `.zshrc`, depending on your shell) afterwards.

[1] Contact the Beam Physics group if you do not have one.

[2] Note that the latest version of Windows also features a bash shell, though it is strongly encouraged to work on either Linux or MacOS

File Structure

In order to generate TRANSOPTR files using xml2optr, the following suggested file and directory structure will make use most straightforward. First, create a working folder in which you intend to store your data.dat and sy.f. In this example, files for the ISAC mass separator (IMS) section are to be created:

```
mkdir ~/optrdev/ims/
mkdir ~/optrdev/ims/tf/
```

The subdirectory /tf/will contain an XML tunefile, which is needed by xml2optr to specify start and end locations along the beamline, in addition to beam properties. The tunefile can optionally contain optics setpoints, if desired, though it must also point to a tune in acc/. An example tunefile, rick.xml, is shown below:

```
<root xmlns:xi='http://www.w3.org/2001/XInclude' path="ite-tm4-sis-ily-yield">

<!--from lin12.triumf.ca/optr/ISAC/Sep-LEBT/matlab/data.dat (rick pointed me there) 2021-08-19-->
<optr s11="0.45*cm" s22="2.0*mrad" s33="0.1*cm" s44="9.0*mrad"
s55="0.0*cm" s66="1e-30*mrad" r12="0.0" r34="0.0" r56="0.0"
bunchcharge='3.0e-9*C'  start="IMS:YSLIT0" end="IMS:YSLIT11B"/>

<tune mass="199.9799*u" chargestate="1.0" energy="60.0*keV"/>

    <set pv="IMS:Q1:POS:VOL"  value="2445.0*V"/>
    <set pv="IMS:Q2:POS:VOL"  value="1933.0*V"/>
    <set pv="IMS:Q3:POS:VOL"  value="2445.0*V"/>

    <set pv="IMS:Q4:POS:VOL"  value="641.3*V"/>
    <set pv="IMS:Q5:POS:VOL"  value="806.7*V"/>
    <set pv="IMS:Q6:POS:VOL"  value="641.3*V"/>

    <set pv="IMS:Q7:POS:VOL"  value="1692.0*V"/>
    <set pv="IMS:Q8:POS:VOL"  value="2441.0*V"/>
    <set pv="IMS:Q9:POS:VOL"  value="3291.0*V"/>
    <set pv="IMS:Q10:POS:VOL"  value="5123.0*V"/>

<xi:include href='isac/tune/ite-tm4-sis-ily-yield_ref.xml'
xpointer="xpointer(//root/tune)" parse="xml"/>
</root>
```

A few notes:

- The <root> tag specifies a path, ite-tm4-sis-ily-yield, which corresponds to an XML path file in acc/, at $MYAPPSDIR/acc/isac/path/ite-tm4-sis-ily-yield.xml. This path file points to sequence files in acc/, needed to construct the entire beampath. In this case, the path starts from ITE (ISAC Target East), Target Module 4, Surface Ionization Source, and goes until the ILY section, containing the ISAC yield station. Path descriptions follow this standard, start to end structure.
- The <optr> tag contains information needed by TRANSOPTR to define initial beam parameters, with for example s11 corresponding to the horizontal beam size, which is the square root of σ-matrix element 11. Likewise, s22 is the beam distribution x-momentum, while r12 is the correlation coefficient between both.
- The start and end of the desired sub-sequence of elements to be included in sy.f are specified within the <optr> tag, and must correspond precisely to the element name as represented in an acc/ sequence file. In this example, the simulation starts at IMS:YSLIT0, located after the ISAC pre-separator magnet, and terminates at IMS:YSLIT11B, located just after the ISAC mass separator magnet.
- bunchcharge gives the integrated bunch charge if TRANSOPTR is run in mode-5, otherwise it is the beam current, required for first order space charge computation, if relevant. If no space charge is desired, the bunch charge can be set to something small, or zero.
- Observe that all entries multiply relevant units, using the $*$ character. This is essential, as the XML tunefile can be specified in any unit of your choosing and will be appropriately converted at xml2optr execution.
- A <tune> tag contains information such as species mass in atomic mass units (u), the ionization state and the starting beam energy.
- <xi:include> points to $MYAPPSDIR/acc/isac/tune/ite-tm4-sis-ily-yield_ref.xml(the $MYAPPSDIR/acc/ part of the must not be specified in this line). This tunefile contains values for all of the quadrupole lenses in that path, however custom values are provided in the <set pv=...> tags, with appropriate units.

Using the above displayed rick.xml, a set of TRANSOPTR files repsresnting the IMS section, from IMS:YSLIT0 up to IMS:YSLIT11B, for a 200u beam, charge state 1+ at energy 60 $keV/$u can be generated programmatically by typing:

```
cd ~/optrdev/ims/
xml2optr -tf tf/rick.xml ite-tm4-sis-ily-yield
```

This generates all necessary files for a TRANSOPTR simulation. Note that by default, data.dat specifies mode-5 use[1], though for IMS, being continuous beams, we want to use mode-3. Thus, we change the second integer in data.dat line 2, from 5 to 3. Running TRANSOPTR, where we've specified the alias:

```
alias optr='$MYAPPSDIR/transoptr/runoptr.sh'
```

we obtain the envelopes for that section, shown in Figure 1. Though this is not the only way in which `xml2optr` can be used, it is the most commonly used for ISAC accelerators and should serve as a basis showcasing how to generate arbitrary TRIUMF beamline and accelerator simulations in `TRANSOPTR`!

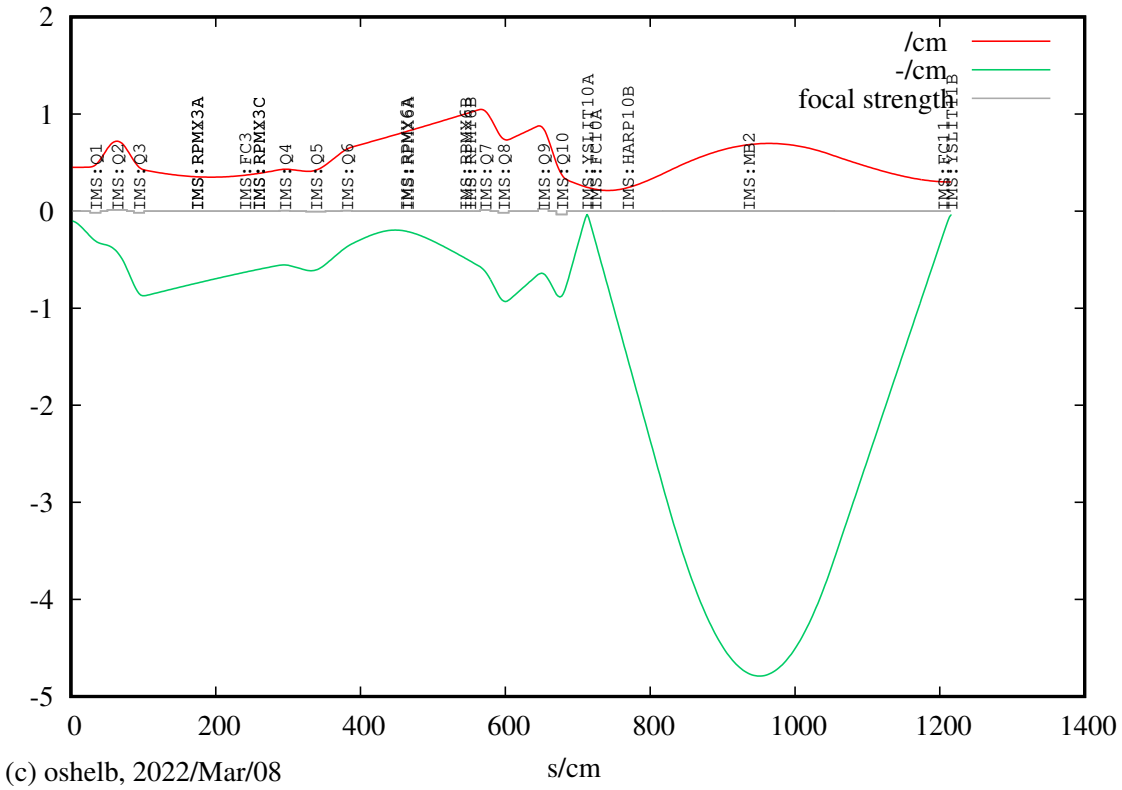

Figure 1: `TRANSOPTR` simulation, automatically generated via `xml2optr` using the tunefile `rick.xml` specified within this document.

References

[1] Olivier Shelbaya. A Quick TRANSOPTR Primer. Technical Report TRI-BN-20-06, TRIUMF, 2020.

Appendix A - Supplementary Example

Example below generates an e-linac set of files using the tune stored in `acc/`:

```
xml2optr -t egun-ehat-fc4_ref.xml -f $MYAPPSDIR/acc/elinac/path/egun-ehat-fc4.xml
```

```
xml2optr -tf $MYAPPSDIR/acc/elinac/tune/egun-ehat-fc4_ref.xml egun-ehat-fc4
```