THE TPSA AND NORMAL FORM ANALYSIS IN TESLA

Lingyun Yang, NSLS-II, BNL, NY, USA

Abstract

TESLA is a single partile dynamics simulation code. In the recent development and following the algorithms in PTC [1], some normal form analysis and complex TPSA have been added to it. The lattice functions calculation based on symplectic integrator and normal form analysis are more general and robust. A Python module is also developed by wrapping the C++ code to make accelerator beam dynamics simulation and data analysis in both interactive and batch mode.

INTRODUCTION

TESLA is a single particle dynamics code. For the modeling of storage ring lattice, it follows the framework set in FPP/PTC [2, 3]. A nonlinear high order transfer map is first produced by truncated power series algebra (TPSA) [4, 5] and analyzed by normal form method to extract the lattice properties.

Althoug using C++ templates, the initial TPSA in TESLA was only instantiated with real numbers [5] to follow the FORTRAN77 implementation of normal form algorithm [6]. In fact, if there was a complex TPSA available in FORTRAN77, the implementation code would be much simpler and cleaner as the theory described in paper [7]. Since the TPSA algorithm in TESLA are general for any element type, complex or real, the only change needs to make is the instantiating type for the template, plus some functions specific to complex numbers, e.g. real, imag, abs, norm, arg, ...

- TPSA operations: +, -, *, /, +=, -=, *=, /=
- TPSA functions: compress, clear, truncate, exp, log, sin, cos, tan, sqrt, asin, acos, atan, pderivative, pbracket.
- TPSMap: *, +, -, substitute, partialInverse, inverse.
- CTPSMap (complex TPSA map): all functions in TPSMap, real, imag, conjugate.

As pointed out by E. Forest, once having a complex TPSA, the implementation of normal form is more clear. There is no need to simulate complex map operations with two real maps. This brings the new normalization routines in PTC and therefore followed by TESLA.

One example of complex TPSA instantiation is shown below:

```
const int NV = 2;
const int ND = 2;
TPST_<complex<double> > x(NV, ND), p(NV, ND);
x.setVariable(0, 0.0);
p.setVariable(1, 0.0);
TPST_<complex<double> > t1(NV, ND);
t1 = 0.5*x*x + p*p;
```

The output, i.e. the coefficient of power series expansion of $t1 = x^2/2 + p^2$ is simply 1/2 and 1 before x^2 and p^2.

```
t1=
 V : D= 2 L= 6           Base [ 6/6 ]
-------------------------------------
(0.000e+00,0.000e+00)     0 0     0
(5.000e-01,0.000e+00)     2 0     3
(1.000e+00,0.000e+00)     0 2     5
```

where $D = 2$ is the highest order, base is the exponent of each dependent variables x and p. The above data means $t1 = 0 * x^0p^0 + 0.5 * x^2p^2 + 1.0 * x^0p^2$. The imaginary part in the coefficients for each term in $t1$ are all zero.

Trignometric functions, derivatives and substitutions are also obtained in the same way as TPST_<double> for complex TPSA.

NORMAL FORM

The algorithm for normal form is described in early publications [2, 3, 6] and implemented in PTC. Here I am outline it briefly how TESLA normalizes the map with the same algorithm, but different TPSA library.

A full turn map is obtained first as $\mathcal{M}_0(z_1, z_2, \ldots, \delta)$. The closed orbit is the constant part of TPSA map. Then the δ-dependent part will be taken out by $\mathcal{M}_0 = \mathcal{A}_0 \circ \mathcal{M}_1 \circ \mathcal{A}_0^{-1}$. From closed orbit condition $z + \delta\eta = \mathcal{M}(z + \delta\eta) + \delta v$, we can solve for η [3],

$$\eta = (1 - \mathcal{M}_z)^{-1} v$$

where $\mathcal{M}_z$ is a map for all z_i, v takes out contributions from all z_i and $\mathcal{A}_0 = z + \delta\eta$. An example of $\mathcal{A}_0$ is

```
 V: D=3 L=56                                                  Base  [ 4/56 ]
-------------------------------------------------------------------------
  1.00e+00  0.00e+00  0.00e+00  0.00e+00   1 0 0 0 0 0    1
  0.00e+00  1.00e+00  0.00e+00  0.00e+00   0 1 0 0 0 0    2
  0.00e+00  0.00e+00  1.00e+00  0.00e+00   0 0 1 0 0 0    3
  0.00e+00  0.00e+00  0.00e+00  1.00e+00   0 0 0 1 0 0    4
  1.56e-03  1.82e-03 -8.76e-03 -8.13e-03   0 0 0 0 0 1    5
  4.96e+00 -1.79e-01  7.06e-01  6.96e-01   0 0 0 0 0 2   20
 -1.20e+02  1.89e+01 -8.77e+01 -8.84e+01   0 0 0 0 0 3   55
```

In this perturbative approach its inverse map is $A_0^{-1} = z - \delta\eta$ with sign reversed for the δ-dependent coefficients.

The linear map is diagonalized by $\mathcal{A}_2$ from the eigenvectors of $\mathcal{M}$. We have chosen the convension $A_2(0, 1) = A(2, 3) = 0$. The current residual map $\mathcal{M}_2$ has a rotation map R in its linear part. An example $\mathcal{A}_2$ is in the following:

```
 V: D= 4 L= 5                                  Base  [ 4 / 126 ]
-------------------------------------------------------------
  2.18e+00  1.39e-12  1.34e-01  3.57e-04  1 0 0 0 0  1
  0.00e+00  4.58e-01 -3.00e-04  1.60e-01  0 1 0 0 0  2
  3.20e-01 -4.91e-05 -9.16e-01  1.42e-12  0 0 1 0 0  3
  2.34e-04  6.73e-02 -3.61e-17 -1.09e+00  0 0 0 1 0  4
```

and the diagonalized map $\mathcal{M}_2$ (shown only up to second order) is

ISBN 978-3-95450-136-6

```
V: D= 4 L= 126                                          Base [ 5/126 ]
-------------------------------------------------------------------
-9.80e-01 -1.98e-01 -1.34e-16 -7.75e-17  0.0  1 0 0 0 0  1
 1.98e-01 -9.80e-01  1.14e-16  2.49e-16  0.0  0 1 0 0 0  2
-1.16e-16  3.14e-18 -9.80e-01 -1.98e-01  0.0  0 0 1 0 0  3
 4.16e-17 -4.71e-16  1.98e-01 -9.80e-01  0.0  0 0 0 1 0  4
-1.42e-23 -2.75e-23  5.56e-23  3.84e-22  1.0  0 0 0 0 1  5
 7.81e-06 -9.77e-05  5.70e-06 -2.08e-05       2 0 0 0 0  6
-1.91e-05  2.01e-05 -4.17e-06 -5.02e-06       1 1 0 0 0  7
 1.23e-05 -4.16e-05 -3.01e-05  3.29e-05       1 0 1 0 0  8
-4.29e-06 -4.34e-06 -5.92e-07  2.32e-05       1 0 0 1 0  9
 3.89e+00 -1.92e+01 -1.29e-04 -7.84e-02       1 0 0 0 1 10
-9.59e-05 -1.19e-05 -1.46e-05 -3.16e-06       0 2 0 0 0 11
 1.22e-06 -3.76e-06 -2.82e-07  2.32e-05       0 1 1 0 0 12
-2.99e-05 -8.33e-07 -6.60e-08 -4.45e-06       0 1 0 1 0 13
 1.91e+01  3.89e+00 -7.84e-02  1.29e-04       0 1 0 0 1 14
-1.50e-05  1.64e-05 -6.71e-06  2.02e-06       0 0 2 0 0 15
-2.92e-07  2.33e-05  3.88e-06  1.29e-05       0 0 1 1 0 16
-4.42e-05 -7.84e-02  3.89e+00 -1.87e+01       0 0 1 0 1 17
-4.31e-08 -2.07e-06 -1.58e-07 -3.20e-06       0 0 0 2 0 18
-7.84e-02  4.42e-05  1.96e+01  3.89e+00       0 0 0 1 1 19
 1.70e-25  2.23e-22 -4.81e-26 -1.51e-21       0 0 0 0 2 20
 ...
```

The linear map has tow rotation matrix at the diagonal and zeros (machine precision) at the off-diagonal. Since at this stage the δ-dependence is removed, the coefficient for base `00001` is identity. If we plot the coordinates by applying $\mathcal{M}_2$ on the initial coordinates iteratively, it is a circle up to the linear order.

The residual nonlinear map becomes $\mathcal{M}_{\rm nl} = \mathcal{M}_2 \circ R^{-1}$, and $\mathcal{M}_0 = \mathcal{A}_0 \circ \mathcal{A}_1 \circ \mathcal{M}_{\rm nl} \circ \mathcal{R} \circ \mathcal{A}_1^{-1} \circ \mathcal{A}_0^{-1}$. The further normalization is done order-by-order in a perturbative approximation. However, depends on what quantities we are interested, we may not need go high order normalizations. A method of calulating some twiss functions from averaging is proposed in Ref. [3], e.g. beta function is an average of x^2. The normalized map makes averaging straight forward.

For the map starting at a different s-location s_2, the full turn map is

$$\mathcal{M}_2 = \mathcal{M}_{1\to 2} \circ \mathcal{M}_1 \circ \mathcal{M}_{2\to 1} \equiv \mathcal{M}_{12} \circ \mathcal{M}_1 \circ \mathcal{M}_{12}^{-1}$$

Once we have normalized $\mathcal{M}_1 = \mathcal{A} \circ \mathcal{M}_{\rm nl} \circ \mathcal{R} \circ \mathcal{A}^{-1}$, then we can have

$$\mathcal{M}_2 = (\mathcal{M}_{12} \circ \mathcal{A}) \circ \mathcal{M}_{\rm nl} \circ \mathcal{R} \circ (\mathcal{M}_{12} \circ \mathcal{A})^{-1}$$

i.e. $\mathcal{B} \equiv \mathcal{M}_{12} \circ \mathcal{A}$ normalized the full turn map $\mathcal{M}_2$ starting at s_2. It simply means if we can transport the map $\mathcal{A}$ from s_1 to s_2, we can normalize the full turn maps starting at any s-locations. In TESLA or any polymorphic code this is trivial, transporting a particle coordinates are same as transporting a map. We know that $\mathcal{R}$ has the tune (rotation) information and $\mathcal{A}$ has the s-dependant β functions. In this way we can get β around the ring.

PYTHON BINDINGS

Python is a high level programming language convenient for both interactive or batch scripting. Wrapping a C/C++ simulation code as Python library is well supported. Such an API is also developed for TESLA. A short Python script which loads the lattice and normalize the one turn map is given in the following:

```
import tesla
ring = tesla.Ring("fodo_02.tslat", "RING")
# six phase space varialbe, 4 independent
m = tesla.TPSMap(5,5,2)
m.resetI()
m.c=[1e-6, 0, 0, 0, 1e-2, 0]
print m
err = ring.trackTPSMap(m, \
    0, ring.elements(), tesla.AP_TRK_DEFAULT)
print m
nf = tesla.NormalForm()
nf.normalize(m, 4)
print nf.A0()
print nf.A1()
```

ACKNOWLEDGEMENT

I want to thank Dr. Samuel Krinsky (1945–2014) for his encouragement, E. Forest for sharing his various notes and FORTRAN code. I am grateful for support from the NSLS-II. This work is supported in part by the U.S. Department of Energy (DOE) under contract No. DE-AC02-98CH1-886.

APPENDIX

A lattice input for TESLA is like the following. The format is close to MAD-8 [8] but in a former grammar.

```
MKBPM: Marker;
QH1G2C30A: Quadrupole, L= 0.275, K1= -0.633;
SQHHG2C30A: Quadrupole, L= 0.1;
QH2G2C30A: Quadrupole, L= 0.448, K1= 1.477;
...
B1G5C01B: Dipole, L= 2.62, ANGLE= 0.1047198,
  E1= 0.05236, E2= 0.05236;
DH02G1A: Drift, L= 4.29379;
...
CELL: LINE=(DH02G1A, ..., DH01G1A);
RING: LINE=(MK0, 15*CELL);
setup, line="RING", shared=false;
#
update, name="B.*", type="Dipole",
  property="SLICE", value=15;
update, name="Q[LHM].*", type="Quad.*",
  property="SLICE", value=10;
update, name="S[LHM].*", type="Sext.*",
  property="SLICE", value=5;

save_lattice, h5group="ring_lat";
basic, h5group = "basic";

frequency_map, nturn=256,
  x=(-0.025, 0.025, 150),
  dp=(-0.03, 0.03, 120), y=1e-5, ppn=20,
  naff_iter=30, h5group="fma_xdp";
```

REFERENCES

[1] E. Forest, Y. Nogiwa, F. Schmidt, “The FPP documentation”, ICAP’06, Chamonix, France, 2006.

[2] E. Forest, “Beam dynamics: a new attitude and framework”, Harwood Academic Publishers, 1998.

[3] E. Forest, “Implementation and Illustration of Perturbation Theory in a Tracking Code”, unpublished.

ISBN 978-3-95450-136-6

[4] M. Berz, "Differential Algebraic Description of Beam Dynamics to Very High Orders", Particle Accelerators, p. 109–124, 1989.

[5] L. Yang, "Array Based Truncated Power Series Package", ICAP'09, p. 371–373, San Francisco, CA, USA.

[6] E. Forest, M. Berz, J. Irwin, "Normal form methods for complicated periodic systems", Particle Accelerators, p. 91–107, 1989.

[7] E. Forest, private communications.

[8] "MAD: Methodical Accelerator Design V8", http://mad.home.cern.ch/mad/mad8web/mad8.html

ISBN 978-3-95450-136-6