dAk

Lingyun Yang, lyyang@lbl.gov

February 25, 2010 (Rev. –revision–)

Contents

1 Introduction

This document serves as user manual for Tracy lattice format and the C implementation of this lattice parser.

For users who only need to read the first part, i.e. section 2. For people who wants to know more about the internal implementation, e.g. if you want to write a converter, please read more sections after the first part.

2 Lattice Format for Users

2.1 Lattice

If you already know the lattice file in MAD-8 or Elegant, you will find Tracy has very similar lattice format. The following code is an example:

```
! an example
c = 3.0e8; ! defines a variable
PUE: BPM;  ! an element
T: MARKER;
title, "This is an example";
PUE: BPM;
QD: quad, L=0.2, K=3.0;
pi = 3.14;
QD.L = 5*sin(pi/4);
QD.K = abs(QD.L - 2.0);
end;
```

2.2 Include a file

You may nest files with `include`:

```
! We suggest put magnets definition into a single file,
! and use the master file to do different task.
! the lattice files are regarded as "static" ones, doesn't need
! to be changed from task to task.
!
! The file "lattice.lat" must be at the current directory you are running
! the program

include, "lattice.lat";

! Do different task in the following.
```

[TODO]: The master file and lattice file should be in same directory where you run your program. This restriction may be relaxed in the future. (After defining a "searching direc

2.2.1 Introduction

The grammar is context free1 and the text is in free-format[1] style.[2]

- Each statement must end with ";". A statement can be a variable definition, an element definition or an action.
- String is enclosed by two ". It can not span over more than one line. It doesn't support the escape character, i.e. ``This is a \"string\"'' is not acceptable.
- Lattice file is not case sensitive.
- Each statement can span muliple lines, "&" can connect these lines but not required. (here for compatibility with MAD-8 lattice, you don't need to remove them when converting MAD-8 lattice into Tracy format).
- arithmetics and frequently used mathematical functions can be used: abs(), sqrt(), sin(), cos(), log(), exp(), arctan().
- A property of magnets can be referred or changed later, as `Element.Property`.
- The definition of element is:

```
element_name: family [, property = expression];
```

2.2.2 Marker

```
! marker has zero length, no property.
MK1: Marker;
```

2.2.3 Beam Position Monitor

```
! BPM is zero length.
PUE: BPM;
```

2.2.4 Drift

```
DL: Drift, L = 2.0;
```

[1] White space (space, tab, and newline) has no significance (except in strings), i.e. can be inserted anywhere between tokens.

[2] No hard limit on the length of each line, or the length of the lattice file.

2.2.5 Dipole(Bend)

L [m]	length ($\rho\phi$)
T [degrees]	bend angle
T1 [degree]	entrance angle
T2 [degree]	exit angle
Gap [m]	gap
K [m^{-2}]	b2 (gradient)
N	no of integration steps
Method	integration method: (0—2—4), default: 2 (matrix style, 2nd and 4th order symplectic integrator)
HOM	list of systematic higher order multipole errors (random errors are assigned in the input file): n, bn, an, (order, integrated skew- and normal multipole strengths);

Example

```
! A bending magnet.
B : Bending, L=0.70, T=10.0, T1:=5.0, T2:=5.0,
K=-1.0, N=8, Method=2;
```

2.2.6 Corrector

L [m]	default: 0 (thin kick)

Example:

```
COH : HCorrector;
COV : VCorrector;
COHV: Corrector;
```

2.2.7 Quadrupole

Quadrupole

L	$[m]$
Tilt3	Design roll angle [degrees]
K $[m^{-2}]$	b2 (gradient)
N	no of integration steps
Method	integration method: (0—2—4), default: 2 (matrix style, 2nd and 4th order symplectic integrator
HOM	list of systematic higher order multipole errors (random errors are assigned in the input file): n, bn, an, (order, integrated skew- and normal multipole strengths)

Example

```
QF : Quadrupole, L=0.5, K=2.2134, N=4, Method=4;
```

2.2.8 Sextupole

L $[m]$	length [m], default 0 (thin kick)
Tilt3	Design roll angle [degrees]
K $[m^{-3}]$	b3 sextupole strength
N	no of integration steps
Method	integration method: 0 — 2 — 4), default: 2 (matrix style, 2nd and 4th order symplectic integrator;
HOM	list of systematic higher order multipole errors (random errors are assigned in the input file): n, bn, an, (order, integrated skew- and normal multipole strengths);

Example

```
SF : Sextupole, K=-10.2363;
```

2.2.9 Multipole

Multipole

L	length [m]
Tilt3	Design roll angle [degrees],]
T	bend angle [degrees]
T1	entrance angle [degrees]
T2	exit angle [degrees]
N	no of integration steps
Method	integration method: 0 — 2 — 4), default: 2 (matrix style, 2nd and 4th order symplectic integrator;
HOM	list of systematic higher order multipole errors (random errors are assigned in the input file): n, bn, an, (order, integrated skew- and normal multipole strengths)

Examples

```
B : multipole, L=0.70, T=10.0, T1=5.0, T2=5.0, HOM=(2, -1.0, 0), N=8, Method=2;
QF : multipole, L=0.70, HOM=(2, 2.50, 0.0, 4, 1.01e7, 0.0), N=8, Method=2;
```

2.2.10 Wiggler

L	length [m]
BoBrho	B/Brho [m-1]
Lambda	period [m]
kx [m]	
N	no of integration steps
Method	integration method: 0 — 2 — 4), default: 2 (matrix style, 2nd and 4th order symplectic integrator;

Example

```
U143: wiggler, L=4.80, BoBrho=0.5, Lambda=0.15, N=20, Method=0;
```

2.2.11 Cavity

Frequency	Frequency [Hz]
Voltage	RF amplitude [V]
HarNum	harmonic number (for absolute path length calculations)

Example

```
CAV : Cavity, Frequency = 499.95e6, Voltage=1.22e6, HarNum=328;
```

2.2.12 Beamline

```
BL: Line=(QD, DL, QF, QF, DL, QD);
RING: Line=(3*BL, -BL, 3*(BL, BL), -2*(BL, BL));
```

2.3 Convert Old Tracy Lattice

- `RING: ELEM1, ELEM2; ---> RING: LINE=(ELEM1, ELEM2);`
- comment out ”define lattice;”
- Use title, ”Title of Lattice”;
- comment out ”CELL: RING, SYYMETRY=1;”
- `"Beam Position Monitor" ---> BPM.`
- `HCM: HCorrector,Method=Meth;`
- `INV(element) ---> -element`

2.4 Example

You may find a long example lattice from source code, it is in `test/example.lat`.

Here is the full lattice of NSLS-II (very draft, only to show the format Tracy accepts):

Listing 1: A Tracy Lattice Example

```
{ define lattice; }
title,"New␣R3L3SBS4-C6x6Qfgdr␣EX_2.002␣TPW␣H(9.3)␣L(6.6M)␣PMTPW(0.40M)";

E0 = 3.0; dP = 1e-6; CODeps = 1e-10; Meth = 4; Nbend = 4; Nquad = 4;
pi = 4.0*arctan(1.0);
```

```
beam, energy = E0;
! P1: Beam Position Monitor ;
! P2: Beam Position Monitor ;
! P3: Beam Position Monitor ;
```

As shown above, you can define your own variable, and use mathematical functions, e.g. sin(), cos(), abs(), sqrt(), log(), exp(), ...

The elements are defined as following:

```
PUE: BPM;

{CL1X: Corrector, Horizontal, Method= Meth;
 ......
 CM1Y: Corrector, Vertical, Method= Meth;}

HCM: HCorrector, Method= Meth;
VCM: VCorrector, Method= Meth;

DH0:  Drift, L = 4.650;

{DL0, DH1A, DH2A, DH2B, DH2C, DH3A, DH3B, DH4A, DH4B, DH4C, DM1A,
DPW, DM1B, DM1C, DM2A, DM2B, DM2C, DL3A, DL3B, DL4A, DL4B, DL1A,
DL2A, DL2B, DL2C, DSX, DSD, DSC, DSL}

QH1 : Quadrupole, L = 0.300000, K = -0.621453, N = Nquad, Method = Meth;

{QH2, QH3, QM2, QM1, QL3, QL2, QL1 }

B1: Bending, L = 2.62, T = 6.0000, T1 =3.000, T2 = 3.0000, N = Nbend, Method = Meth;

SL1: Sextupole, K =  1.996040/0.2, L = 0.2, N = 1, Method = Meth;

{SL2, SL3, SM1, SM2, SH1, SH2, SH3, SH4}

SQ: Multipole, N = 1, Method = Meth;

MP: Marker; SS: Marker; LS: Marker; SEPTUM: Marker;
GS: Multipole, N = 1, Method = Meth;
GE: Multipole, N = 1, Method = Meth;
```

Please remember do not use keyword as your variable name. i.e. `L=2.0; QD: Quad, L=L;`.

```
c0 = 2.99792458e8;  h_rf = 1300;  C = 791.958;  Brho = 1e9*E0/c0;

{ Mini Gap Undulator CPMU or IVU -- change lambda to 20 mm?}
lambda_CPMU = 19e-3;
n_CPMU = 160;
B_CPMU = 2.2/(0.934*1e2*lambda_CPMU);
L_CPMU = n_CPMU*lambda_CPMU;

D_CPMU: Drift, L = -L_CPMU/2.0;

CPMU1: Wiggler, L = L_CPMU/2.0, lambda = lambda_CPMU,
       kxV = 0.0, BoBrhoV = B_CPMU/Brho, N = n_CPMU*4, Method = 1,
```

```
        harm=(1, 0.0, B_CPMU/Brho, 0, 0, 0);

CPMU: LINE=(D_CPMU, CPMU1);

{<name>: Wiggler,  L       = <length [m]>,
         BoBrhoV = <B/Brho [1/m]>,
         BoBrhoH = <B/Brho [1/m]>,
         Lambda  = <period [m]>,
         kxV     = <[m]>,
         kxH     = <[m]>,
         phi     = <phase [deg]>,
         harm(n, kxV, BoBrhoV, kxH, BoBrhoH, phi)
         ...
         N       = <no of integration steps>,
         Method  = <method>
}
! Example
!
! U143: wiggler, L=4.80, K=0.5, Lambda=0.15, N=20, Method=0;
! EPU:  wiggler, L=4.80, Lambda=0.15, N=20, Method=0,
!       harm=(3, kxV_3, BoBrhoV_3, kxH_3, BoBrhoH_3, phi_3,
!             ...
!            5, kxV_5, BoBrhoV_5, kxH_5, BoBrhoH_5, phi_5);

{ Damping wiggler }
lambda_DW = 9e-2; n_DW = 88; L_DW = n_DW*lambda_DW; B_DW=1.81246*1;

D_DW: Drift, L = -L_DW/2.0;

{DW1: Wiggler, L = L_DW/2.0, lambda = lambda_DW, kxV = 0.0,
      BoBrhoV = B_DW/Brho, N = n_DW*4, Method = 1;}
{Toshi's standard DW poles}
DW1: Wiggler, L = L_DW/2.0, lambda = lambda_DW, kxV = 5.0341*0,
     BoBrhoV = B_DW/Brho, N = n_DW*5, Method = 1,
     harm = (1,  5.0341*0,  B_DW/Brho, 0, 0, 0,
             3,  1.9917*1,  0.044716/Brho*1, 0, 0, 0,
             5,  1.0064*1, -0.060881/Brho*1, 0, 0, 0,
             7,  1.0000*1, -0.007070/Brho*1, 0, 0, 0,
             9,  0.9996*1,  0.004144/Brho*1, 0, 0, 0);

DW: LINE=(D_DW, DW1);

{EPU}
lambda_EPU = 19e-3; n_EPU = 158;
B_EPU = 1.0*2.2/(0.934*1e2*lambda_EPU);
L_EPU = n_EPU*lambda_EPU;
D_EPU: Drift, L = -L_EPU/2.0;
EPU1: Wiggler, L = L_EPU/2.0, lambda = lambda_EPU, kxV = 0.0, kxH = 0.0,
      BoBrhoV = B_EPU/Brho*1.0, BoBrhoH = B_EPU/Brho*1.0, phi=1.571, N = n_EPU*4, Method = 1;
EPU: LINE=(D_EPU, EPU1);

{V_RF = 2.5 MV for 0 DW, 3.1 MV for 3, 3.9 for 8}
CAV: Cavity, Frequency = c0/C*h_rf, Voltage = 3.1e6, Harnum = h_rf;

LB: LINE=(SL1,DL1A,PUE,QL1,DL2A,HCM,VCM,DL2B,SL2,DL2C,QL2,DL3A,SL3,DL3B,QL3,
          PUE,DL4A,HCM,VCM);
HB: LINE=(SH1,DH1A,PUE,QH1,DH2A,HCM,VCM,DH2B,SH2,DH2C,QH2,DH3A,SH3,
          DH3B,QH3,PUE,DH4A,SH4,DH4B,HCM,VCM);
```

```
HB_SQ: LINE=(SH1,DH1A,PUE,QH1,DH2A,HCM,VCM,SQ,DH2B,SH2,DH2C,QH2,DH3A,
    SH3,DH3B,QH3,PUE,DH4A,SH4,DH4B,HCM,VCM);

DISP:       LINE=(DM1B,HCM,VCM,DM1C,PUE,QM1,DM2A,SM1,DM2B,QM2,DM2C);
DISP_SQ:    LINE=(DM1B,HCM,VCM,SQ,DM1C,PUE,QM1,DM2A,SM1,DM2B,QM2,DM2C);
DBA:        LINE=(GS,B1,DM1A,DPW,GE,GS,DISP,SM2,MP,-DISP,GE,GS,DPW,DM1A,B1,GE);
DBA_SQ:     LINE=(GS,B1,DM1A,DPW,GE,GS,DISP_SQ,SM2,MP,-DISP,GE,GS,DPW,DM1A,B1,GE);
ONESP:      LINE=(LS,DH0,GE,GS,HB_SQ,GE,DH4C,DBA_SQ,DL4B,GS,-LB,GE,GS,DL0,SS,DL0,GE,
                  GS,LB,GE,DL4B,DBA,DH4C,GS,-HB,GE,GS,DH0);
ONESP_DW1:  LINE=(LS,DH0,GE,GS,HB_SQ,GE,DH4C,DBA_SQ,DL4B,GS,-LB,GE,GS,DL0,SS,
                  DL0,GE,GS,LB,GE,DL4B,DBA,DH4C,GS,-HB,GE,GS,DH0, DW);
ONESP_CPMU: LINE=(LS,DH0,GE,GS,HB_SQ,GE,DH4C,DBA_SQ,DL4B,GS,-LB,GE,GS,DL0,SS,
                  CPMU,-CPMU,DL0,GE,GS,LB,GE,DL4B,DBA,DH4C,GS,-HB,GE,GS,DH0);
ONESP_EPU:  LINE=(LS,DH0,GE,GS,HB_SQ,GE,DH4C,DBA_SQ,DL4B,GS,-LB,GE,GS,DL0,SS,
                  EPU,-EPU,DL0,GE,GS,LB,GE,DL4B,DBA,DH4C,GS,-HB,GE,GS,DH0);

RING: LINE=(SEPTUM, 3*(ONESP_DW1, -DW),3*ONESP_CPMU, 9*ONESP, CAV);

END;
```

3 Developers

The parser is generated by Lex and Yacc, a standard way in compiler design. Lex scan the lattice, and give each ”word” an token, e.g. *quad* or *Quadrupole* are given a token QUAD, since it is a keyword. For the following lattice

```
title, "A␣Token␣Test";
len = 3.1415;
QD: Quad, L=len, K = len/2.0;
```

Lex will give us

```
TITLE ',' STRING ';'
ID '=' NUMBER ';'
ID ':' QUAD ',' MAG_PROPERTY '=' ID ',' MAG_PROPERTY '=' ID '/' NUMBER ';'
```

together with the token name, we should also return an value with them, since we may latter use this value to do calculation. e.g. the ”len” in ”L=len”.

Yacc will rely on the return of Lex, and do the grammar matching.

```
statement: ID ':' QD ',' property_list ';' { /* our action */}
property_list: property {}
             | property_list property {}
property: MAG_PROPERTY '=' expression {}
expression: ....... {}
```

for the quadrupole definition, we can match it as the above grammar. Once it is matched, we do the data processing, assign the data in these token to our internal data structure.

3.1 Scanner

A few notes about scanner:

- start from the *union* in parser. It defines the data pass from scanner to parser.
- When evaluating an assignment, with symbols in right hand side, we need both its name and value(which is usually defined several lines before current point). Therefore, we keep a pointer to a structure in *union*. This `symb_tab` keeps all the symbol used for arithmetic calculation.
- A global link table is for arithmetic calculation. Essentially it is a mini calculator, similar to Lex&Yacc 2nd Edition.
- The global link table is freed after the whole scanning and parsing.

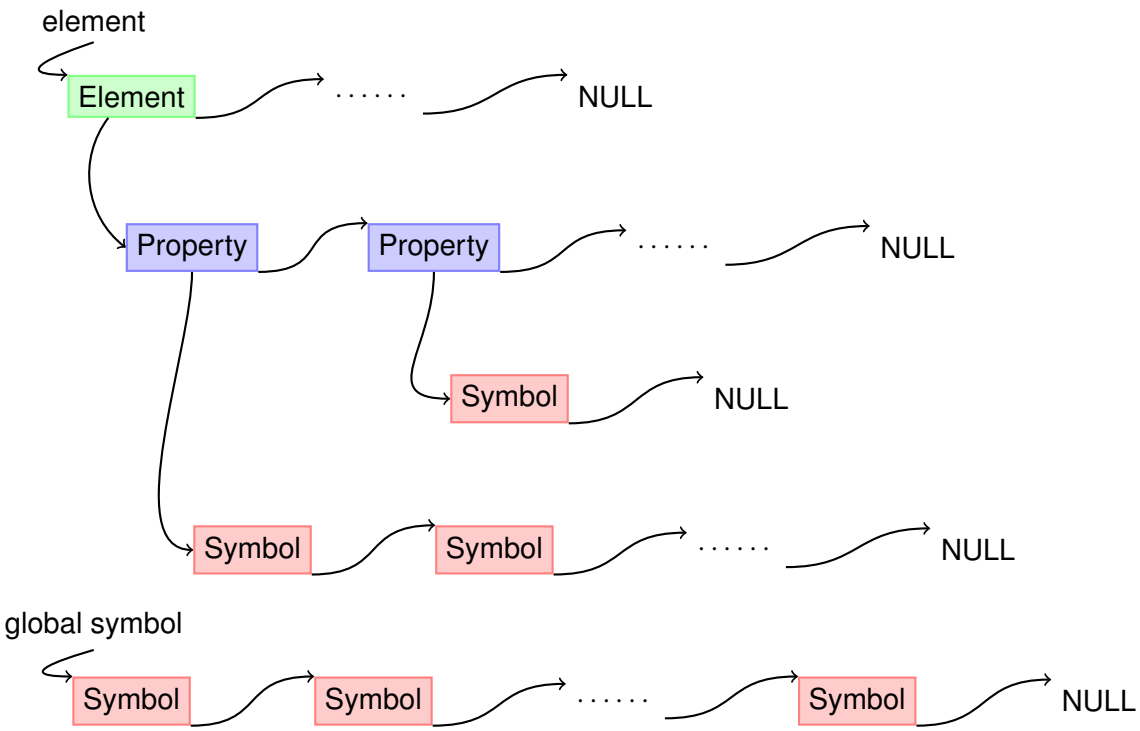
3.2 Parser Grammar

```
- statement: ID ':' Family ',' property_list ';'
- property_list : property_list ',' property
- property : MAG_PROPERTY '=' expression
- property : MAG_PROPERTY '=' expression_list
- expression_list: '(' expression_list ',' expression ')'
```

3.3 Data Structure

The parser is generated by Lex/Yacc.

- lex(flex) scan the lattice, return token: *scanner.l*.
- yacc(bison) organize the grammar: *parser.y*.
- *lattice.h* defines the data structure used by scanner and parser. It also defines the interface to Tracy tracking code.
- *lattice.cc* is a very simple code, call `parse_lattice_flat(char* f, char* flat)` to do the parse.



The property are freed after saving data in element data structure.

3.4 Machine File

Marker

```
name, family num, kid num, element num
type code(-1), integration method, num of integration steps
apertures Xmin, xmax, ymin, ymax
```

Drift

```
name, family num, kid num, element num
type code(-1), integration method, num of integration steps
apertures Xmin, xmax, ymin, ymax
L
```

Multipole

The HOMmax data are calculated, more orders may added after reading the lattice file.

```
name, family num, kid num, element num
type code(-1), integration method, num of integration steps
apertures Xmin, xmax, ymin, ymax
dx, dy, dtheta(design), dtheta(error)
L, Pirho, PTx1, PTx2, Pgap
num of non zero multipole coeff
idx, HOMmax+i, HOMmax-i
...
```

Cavity

```
name, family num, kid num, element num
type code(-1), integration method, num of integration steps
apertures Xmin, xmax, ymin, ymax
vrf/(1e9*energy), 2pi*frf/c0, h, 1e9*energy
```

Thin Kick

```
name, family num, kid num, element num
type code(-1), integration method, num of integration steps
apertures Xmin, xmax, ymin, ymax
```

Wiggler

```
name, family num, kid num, element num
type code(-1), integration method, num of integration steps
apertures Xmin, xmax, ymin, ymax
num of harm
harm, kxV, BoBrhoV, kxH, BOBrhoH, phi
...
```

4 TODO

- vector assignment, `x=(1, 2, 4-1, 2*2);`
- make wiggler definition more clear, combine first harm with higher ones.
- more actions. such as, `show,element=(QD, QF, DL), line=(RING, CELL1, CELL2);`