TRACY, A Tool for Accelerator Design and Analysis*

Hiroshi Nishimura
Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720, U.S.A.

Introduction

A simulation code TRACY[1] has been developed for accelerator design and analysis. The code can be used for lattice design work, simulation of magnet misalignments, closed orbit calculations and corrections, undulator calculations and particle tracking. TRACY has been used extensively for single particle simulations for the Advanced Light Source(ALS)[2], a 1-2 GeV Synchrotron Radiation Source now under construction at Lawrence Berkeley Laboratory.

Motivation and Philosophy

The physics requirement for making such a code is as follows:
A third-generation, low emittance electron storage ring, optimized for use with undulators, requires strong quadrupoles. This leads to a large natural chromaticity that must be compensated by strong sextupoles. At the same time, misalignments of the required strong quadrupoles produce large closed-orbit distortion around the ring. That is, the lattice becomes very sensitive to magnet misalignments. Finally, closed-orbit distortions at the sextupoles give quadrupole components, which change the betatron functions and, as a result, reduce the dynamic aperture considerably.

A third-generation, low emittance storage ring, whose ideal dynamic aperture is relatively small, has a strong correlation between closed-orbit distortion and dynamic aperture. Therefore, an effective closed-orbit correction capability is crucial to maintaining the performance of the perfect lattice as much as possible. Indeed, the performance of the lattice must be evaluated including the capabilities of its orbit-control system, so the study of closed-orbit distortion and its correction, and that of the tracking for dynamic aperture, have to be carried out simultaneously. The tracking code RACETRACK[3], which has the capability of closed-orbit correction and particle tracking, has been used in the design work of the Advanced Light Source.

To serve as part of the accelerator control system, a practical orbit-control scheme based on a simulation must be developed. A simulation can be thought of as a study of the realistic model of a storage ring, including all the previous issues. The required specification for a simulation code is quite different from that for an ordinary lattice or tracking code. With a lattice or tracking code, it is possible to prepare predefined algorithms that a user can select and combine via input parameters. On the other hand, a simulation code has to accept various kinds of "logic" to perform flexibly; it is impossible to prepare all the possible procedures.

There are two types of input to a simulation code: the lattice definition and the logic definition. The latter should be flexible and has the flavor of a language, which can be done either by creating a new "language" or by adopting some existing language structure as a skeleton. Our solution was to utilize Pascal grammar as the input language for the simulation logic. TRACY is a simulation code wrapped by a Pascal-S compiler[4]. It is essentially a one-pass compiler dedicated to accelerator single-particle simulations. The input file is nothing but a Pascal program, so a user can define and use variables, arrays, procedures and functions. This means that, while TRACY is already equipped with many procedures/functions for a simulation, a user still can define his own algorithm. TRACY input is a "language" and there is no limit, by definition, to the capability of accepting a user's logic.

* Work supported by the Department of Energy under contract No. DE-AC03-76SF00098.

An accelerator physics simulation itself is not simple, and the inside of a Pascal compiler is seemingly very complex, so it may sound a tedious task to combine them. But it is very straightforward to implement new procedures and functions in a compiler. The biggest merit of such an approach is that each procedure/function can stay fairly independent. Thus we can maintain modularity. The reason for this is simple. There is almost no need to establish an invisible connection between modules. Each module is dedicated to only one task, and the relation among modules is programmed later by making use of the language structure of Pascal. This fact has made development and maintenance of TRACY relatively easy.

There are other merits to this approach. The meta-language with which TRACY is coded and the language that TRACY accepts are the same. This means that a user can try his logic by programming a TRACY input file, and, when the logic has been established and found to be useful, it can be incorporated into TRACY directly. A procedure developed in this way will have modularity as mentioned above. Such modularity is very useful when developing routines for practical control purposes. As an example, the kernel of the control system for the ALS is written in the PL/M language (whose compatibility with Pascal is very good), making it easy to port modules to the control system itself. TRACY then acts to test the performance of the control modules.

A Simulation Code TRACY

TRACY is a simulation code using a Pascal compiler as a user interface. The first version was developed on a VAX/VMS system.

Physics of Modeling

The exact Hamiltonian describing the motion of a particle in a ring with ideal sector bending magnets is

$$H=-(1+\frac{x}{\rho})\sqrt{(1+\delta)^2-p_x^2-p_y^2}+\frac{x}{\rho}+\frac{x^2}{2\rho^2}-\frac{k}{2}(x^2-y^2)$$

where $\delta = \delta p/p_0$, p_0=design momentum, and p_x and p_y are also normalized by p_0. TRACY uses the linearized part of H:

$$H_0=-\frac{(p_x^2+p_y^2)}{2(1+\delta)}+\frac{x^2}{2\rho^2}-\frac{k}{2}(x^2-y^2)-\delta\frac{x}{\rho}$$

but has options to correct the error caused by taking H_0 by inserting kicks corresponding to $\delta H=H-H_0$. TRACY also has options to adopt fringe fields that satisfy Maxwell's equations for bending and quadrupole magnets to first order in their strength.[5] In particular, the fringe field of quadrupoles can have a strong contribution to the second-order chromaticity and amplitude-dependent tune shift.

When TRACY uses only H_0 (for an ideal lattice), the results of first- and second-order calculations agree with the tracking code FASTRAC[6], which uses the same Hamiltonian and the same symplectic integrator. (To include nonlinear field errors, which are not treated in TRACY, we must use a more sophisticated code like RACETRACK or FASTRAC.) The approximation of taking H_0 only is valid for large rings but not for small rings, especially in calculating the chromaticity. The effect of fringe fields on the chromaticity is also significiant for small rings. One example of this is PSR[7], the Proton Storage Ring at Los Alamos National Laboratory, which has a dipole bending angle of 36°. In this case, the natural horizontal chromaticity given by H_0 is about -2, though the exact value is about -1. Turning the options for δH and fringe fields on, TRACY calculates the correct chromaticity.

For a rectangular bending magnet, TRACY uses at present the "standard" formula for edge-focusing obtained by putting thin quadrupoles on both sides of a sector bending magnet; this approach gives the wrong chromatic properties. The correct treatment of a rectangular magnet, using Healy's modular approach[8], will be implemented soon.

We can check the validity of the model by turning these options on and off, and comparing the results with an analytic code such as MARYLIE[9]. For practical purposes, such as a simulation of a closed-orbit correction scheme, there is no need to utilize such options. The point is that we can use TRACY for calculations ranging from theoretical investigations to practical simulations for control.

Definition of a Lattice

TRACY reads a lattice definition file and sets up a cell structure that is separate from the program file. TRACY can handle a set of the following elements to define a cell: drift spaces, bending magnets, quadrupole magnets, sextupole magnets, thin dipole "kicks", beam position monitors and steering magnets. Thin dipoles are used to simulate magnet misalignments of quadrupoles, linear field errors and roll angle errors of dipoles. A hard-edge model of a wiggler/undulator is also implemented.

A cell is defined as a series of blocks, where a block is a series of elements. The identifiers used as element names in this file are automatically declared as variables to be used in a program. For example, if we define a quadrupole Q1 as:

```
Q1: focusing quadrupole, L=0.15, K=1.7734;
```

then we can use Q1 to identify this magnet and real variable KQ1 for its strength in a program. The strength of any element can be monitored and modified freely via GetKvalue/SetKvalue procedures in a program. For example, the sequence below reduces the strength of Q1 by 2 %:

```
GetKvalue(Q1,KQ1); { monitor the strength of Q1 }
KQ1:=KQ1*0.98; { reduce it by 2 % }
SetKvalue(Q1,KQ1); { set the strength of Q1 }.
```

Definition of Simulation Algorithm

The other input file, defining the simulation algorithm, is called a program file. It is simply a Pascal program, as shown in examples below. A table of elements and a cell are created from a lattice definition file and are manipulated as programmed in this file. A user can define variables, procedures and functions making use of predefined variables, procedures and functions dedicated to a single particle simulation. The extended features of TRACY as a Pascal compiler are as follows:

Manipulation of Elements: Get and set element in a ring. Get and set parameters of an element.

Parameters of a Cell: Twiss parameters, closed-orbit positions and beam positions can be monitored.

Betatron Tune: Assign a pair of quadrupoles for tune fitting, fit tune, get tune, get amplitude-dependent tune shift and get momentum-dependent tune shift.

Chromaticity: Assign a pair of sextupoles for chromaticity fitting, fit and get chromaticity.

Dispersion: Assign a quadrupole for dispersion fitting, fit and get dispersion at a specified point in a ring.

Synchrotron Integrals: Calculate synchrotron integrals I1,I2,I3,I4, I5, and related quantities, such as emittance and momentum spread.

Closed Orbit: Simulate random misalignments of bending and quadrupole magnets, field and roll angle error of bending magnets. Calculate a closed orbit and set values for beam position monitors. Correct a closed orbit by one of the predefined methods (local-bump or most-effective-corrector method). Get and set the strength of a steering magnet.

Tracking: Track particle for given initial conditions. Get dynamic apertures by using one of the predefined tracking routines. Get phase space distribution. Get Fourier components of particle motion observed at one point in a cell.

Undulator/Wiggler: A hard-edge model is supported. A wiggler is defined in a lattice definition file as follows:

```
U0A : wiggler, L=4.8, N=96, B0=0.8;{T}
```

and a user can change the strength B0 in a program as:

```
SetWiggler(U0A, 0.5);{Set the field of U0A to 0.5 Tesla }.
```

Examples

Closed-Orbit Correction

The first example use of TRACY we show is to simulate random magnet errors, find the closed-orbit distortion and correct it. The lattice is the ALS 1.5-GeV Storage Ring[2]. The calculation is carried out for 100 sets of random statistics.

```
---- Input Program File ----
program COD(ALS12.lat);{<-- assign lattice def. file }
  const  dX=0.15;{ misalignment in [mm] }
         dT=0.50;{ roll angle in [mrad] }
         dB=0.10;{ field error in [%]  }
  var  Xm,Xa,Xr, Ym,Ya,Yr:real;  i,j:integer;
begin
  GetCellFnc(0);
  InitBUMP(0.2,0.1); { init COD correction routine }
  SetCODparm(1,1,100);
  SetGvalue( 0.1, CODdXdy);
  SetGvalue(1.0E-6, CODeps);
  SetRanCut(2.0);
  InitRand(12345);{ initialize random number }
  writeln('        Before          After');
  writeln('X [mm]  Y [mm]   X [mm]  Y [mm]');
  for i:=1 to 100 do
    begin
      NewSeed;{ get new seed for random number}
      ClearCOD;ClearCOH;ClearCOV;{init COD stuff}
      Qerror(QF,KQFH,KQFV ,dX,dX);{errors for QF}
      Qerror(QD,KQDH,KQDV,dX,dX);{errors for QD}
      Berror(BU,KBH,KBV,dX,dX,dT,dB);{errors for bend}
      { COD before correction}
      GetCOD(0.0,0);GetCODstat(Xm,Xa,Xr, Ym,Ya,Yr);
      write(Xr:10:5,Yr:10:5);
      for j:=1 to 50 do ExecBUMP(2.0,1);{COD correction }
      { COD after correction}
      GetCOD(0.0,0);GetCODstat(Xm,Xa,Xr, Ym,Ya,Yr);
      writeln(Xr:10:5,Yr:10:5);
    end;
end.
```

Figure 1 shows the result of this calculation. The closed-orbi correction modules are evaluated this way in TRACY and will b ported to the online control system. To get the distribution c emittance, insert the following statements where required:

```
GetCellFnc(1); { reconstruct a cell }
GetSynch; { carry out synchrotron integrals }
GetGvalue(E,Emittance);. { get emittance }
```

It is also possible to insert a tracking routine to get dynamic aperture.

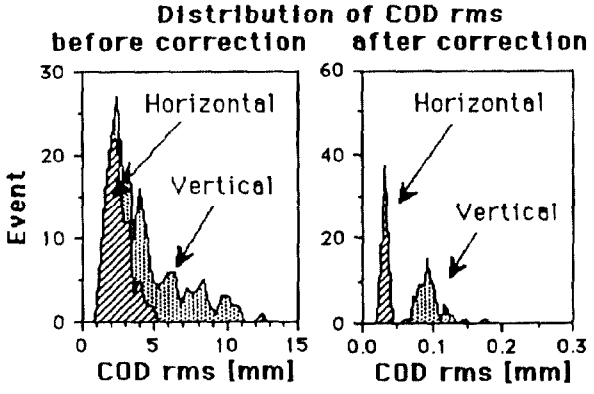

Figure 1. Distribution of Closed Orbit Distortions.

Tune Fitting and Tracking

The next example TRACY calculation involves betatron tune fitting followed by tracking. The lattice is the ALS 1.5 GeV Storage Ring. Changing the vertical tune from 8.05 to 8.95 by 0.01, a chromaticity correction is applied and a vertical dynamic aperture at infinite coupling is calculated by using 1 particle and 100 turns of tracking.

```
----- Input Program File ----
program DynapV(ALS0.LAT);{<-- assign lattice def. file }
    var  A,X,Y,E,Nux,Nuy:real;
      i:integer;
begin
    GetCellFnc(0);
    Nux:=14.27696;
    writeln(' Nuy   QF   QD  Emittance Y[mm]');
    for i:=5 to 95 do
      begin
        Nuy:=8+0.01*i;
        FitTune(Nux,Nuy);{ Tune Fitting }
        GetKvalue(QF,KQF);{ K of QF }
        GetKvalue(QD,KQD);{ K of QD }
        FitChrom(0.0,0.0);{ Chromaticity Correction}
        GetCellFnc(1);{ Generate Cell }
        GetSynch;{ Calculate Synchrotron Integerals }
        GetGvalue(E,Emittance);{ Get Emittance }
        SetUpTrack;{ Set up Traking Routine }
        GetDynap(1.0E8,1,100,2.0,0.1, a,x,y);
        { Dynamic Aperture, 1.0E8 coupling, 1 particle,
          100 turns, scan/initial step=2 mm, eps=0.1 mm}
        writeln(Nuy:7:3,KQF:7:4, KQD:7:4,E:10,Y:6:1);
      end;
end.
---- Output ----
```

Nuy	QF	QD	Emittance	Y[mm]
8.05	2.2083	1.2413	4.084E-09	15.1
8.06	2.2086	1.2422	4.083E-09	15.4
....	....	....	....	....
8.95	2.2319	1.3218	4.095E-09	8.9

Figure 2, based on this result, shows the vertical tune dependence of the vertical dynamic aperture. There are several resonances recognized, such as $3\nu_x+2\nu_y=12\times5$ and $-\nu_x+3\nu_y=12$.

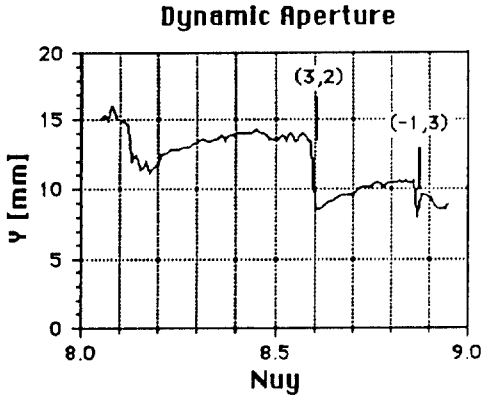

Figure 2. Vertical Tune Dependence of the Vertical Aperture

Undulator/Wiggler

TRACY uses a linearized hard-edge model for undulators/wigglers. This will not be sufficient to determine the dynamic aperture in the presence of an undulator that is strongly nonlinear, though tracking is always possible. However, the hard-edge model is enough to calculate the effects of the device on betatron tune, emittance, momentum compaction, etc. Nonlinear effects will be included in TRACY soon.

Discussion / Conclusion

Our approach of adopting a language structure in order to accept a user-defined algorithm turns out to be very effective and flexible. We recognize that there is an alternative approach to this problem, namely to create a library for simulation, like a graphics library for drawing, and link it to user routines. In this case, there is a non-negligible overhead in linking a large library. This overhead will become larger as the library evolves to have more capabilities. The merit of our method is that the 'overhead of the simulation kernel' has been already compiled and linked. Furthermore, the meta-language is the same as the programming language, so that user defined procedure can be easily integrated into the system. Aside from the execution time of a simulation code, the time required to edit input/output files is important. Therefore the capability of accepting a flexible input format that includes the simulation logic is crucial. A precompiled-and-linked kernel embedded in an existing language is one of the most efficient solutions.

Acknowledgments

The author would like to express his thanks to all the members of the Accelerator Sytems Group of the Advanced Light Source Project at LBL. Special thanks are due to the first two users of TRACY, Dr. A. Jackson and Dr. M. Zisman, who gave much valuable advice and discussion, to Dr. M. Cornacchia and Dr. S. Chattopadhyay for their encouragement and to Dr. E. Forest for his invaluable advice.

References

1) H.Nishimura, "TRACY USERS MANUAL", LBL Report No. LBL-24624. to be published.
2) "1-2 GeV Synchrotron Radiation Source, Conceptional Design Report", July 1986, LBL/PUB-5172 Rev.
3) A.Wrulich, DESY 84-026 (1984).
4) R.E.Berry,"Programming Language Translation", Ellis Horwood Limited, England, 1981.
5) E.Forest and J.Milutinovic,SSC-142, 1988.
6) B.T.Leemann and E.Forest, SSC-133, 1987.
7) A.J.Dragt, Particle Accel. 12, 205, 1982.
8) E.Forest, SSC-141, 1988.
9) A.J.Dragt et al., "MARYLIE 3.0, A Program for Charged Particle Beam Transport Based on Lie Algebraic Methods", Univ. of Maryland, 1987.