Note:INCOMPLETE DRAFT (also outdated)

Tracy-2 User's Manual

J. Bengtsson

Acknowlegdement

This code has its origin in an idea initially realized by H. Nishimura, i.e. to use a standard programming language as the command language for a tracking code. In particular, startingfrom N. Wirth's Pascal-S compiler/interpreter (a strict subsetof Pascal), in collaboration with E. Forest, the standard proceduresand functions of Pascal were enhanced to include routines forbeam dynamics. However, the initial code was developed for theshort terms needs in the lattice design of the Advanced LightSource (ALS) at LBL. The code therefore finally reached a statewhere it could hardly be maintained or modified. The currentcode is a compromize (e.g. Pascal is still used rather than e.g.C or C++) that empasizes generality and flexibility in the userinterface, and is built from the ideas and experiences gainedfrom the earlier codes. One working constraint has been to keepbackwards compability. However, this has been sacrificed in caseswhere generality or flexibility would have to be compromized. It grew out from an initial effort for an on-line model for theAdvanced Light Source (ALS) but finally found little use due toan overall lack of systematic approach in the commissioning process. I would like to thank E. Forest for continued guidance concerningthe single particle dynamics and I am also very grateful to S.Chattopadhyay head of the Center of Beam Physics for his continuedsupport during this work.

Design goals

A clean and straightforward implementation of a magnet. It isdefined by the coefficients in the multipole expansion. Mis-alignmentsare implemented by applying a Euclidian transformation at theentrance and exit of each element. All quantities (length, multipolecomponents, mis-alignments and linear lattice functions) for amagnet can be accessed and modified from the input file.

Choice of integrator: matrix style and 2:nd or 4:th order symplecticintegrator.

Magnets can be referenced induvidually or as families.

Non-linear optimizer (downhill simplex) and singular value decomposition.

Extensions: text files, include files, passing of arrays and recordsto physics routines (on the Pascal-S stack), access to all latticeparameters through the record structure of a magnet, matrix calculations,line numbers.

Linear DA library compatible to simplify possible relinking toM. Berz general DA library.

Compiled input files.

Correction of a few Pascal-S bugs.

Emphasis has been put on implementing generic routines at thePascal-S level and high level routines as include files.

Emphasize on structured and generic compact code.

Graphics: subset of GKS.

The Hamiltonian

The Hamiltonian describing the motion of a charged particlearound a reference trajectory in an external magnetic field isgiven by

$$H_1 \equiv - p_s = - (1 + h_{ref}\, x) \left[\frac{q}{p_0} A_s + \sqrt{1 - \frac{2}{\beta} p_t + p_t^2 - \left(p_x - \frac{q}{p_0} A_x\right)^2 - \left(p_y - \frac{q}{p_0} A_y\right)^2} \right] - \frac{1}{\beta} p_t$$

where

$$p_t \equiv - \frac{E - E_0}{p_0\, c}, \quad h_{ref} = \frac{1}{\rho_{ref}}$$

and

$$\overline{B} = \nabla \times \overline{A}$$

β is the relativistic factor and we are using the phase space coordinates $(x, p_x, y, p_y, ct, p_t)$.These are deviations from a particle following the referencetrajectory with the local curvature h_{ref} and energy E_0. In this curvilinear system[Bengtsson]

$$B_x = \frac{1}{1 + h_{ref}\, x} \frac{\check{Z} A_y}{\check{Z} s} - \frac{\check{Z} A_s}{\check{Z} y}$$
$$B_y = \frac{h_{ref}}{1 + h_{ref}\, x} A_s + \frac{\check{Z} A_s}{\check{Z} x} - \frac{1}{1 + h_{ref}\, x} \frac{\check{Z} A_x}{\check{Z} s}$$
$$B_s = \frac{\check{Z} A_x}{\check{Z} y} - \frac{\check{Z} A_y}{\check{Z} x}$$

Introducing the canonical transformation

$$F_2 = \frac{1}{\beta}\left[1 - \sqrt{1 + \beta^2 \left(2\,\delta + \delta^2\right)}\right]\left(c\,t + \frac{s}{\beta}\right) + \delta\, s, \quad H_2 = H_1 + \frac{\dot{Z} F_2}{\check{Z} s},$$

$$- c\,T = \frac{\check{Z} F_2}{\check{Z} \delta} = - \frac{\beta\,(1 + \delta)}{\sqrt{1 + \beta^2 \left(2\,\delta + \delta^2\right)}}\left(c\,t + \frac{s}{\beta}\right) + s, \qquad p_t = \frac{\check{Z} F_2}{\check{Z}(ct)} = \frac{1}{\beta}\left[1 - \sqrt{1 + \beta^2 \left(2\,\delta + \delta^2\right)}\right]$$

and

$$\delta \equiv \frac{p - p_0}{p_0}$$

where p_0 is the momentum of the referenceparticle. We obtain

$$H_2 = -(1 + h_{ref}\, x)\left[\frac{q}{p_0} A_s + \sqrt{(1+\delta)^2 - \left(p_x - \frac{q}{p_0} A_x\right)^2 - \left(p_y - \frac{q}{p_0} A_y\right)^2}\right] + \delta$$

using the phase space coordinates $(x,\ p_x,\ y,\ p_y,\ -cT,\ \delta)$. Note that T is not the time of flight t which is given by

$$c\,t = \frac{1}{\beta}\left[\frac{\sqrt{1 + \beta^2\left(2\,\delta + \delta^2\right)}}{1+\delta}\,(c\,T + s) - s\right]$$

We will only consider the ultra-relativistic limit for which

$$p_t \to -\,\delta\,, \quad c\,t \to c\,T \quad \text{when} \quad \beta \to 1$$

It is straightforward to generalize if this approximation is notvalid.

In the case of a sectorbend we have

$$\frac{q}{p_0} A_s = -\frac{1}{2}(1 + h_B\, x)$$

We linearize the equations of motion by expanding the Hamiltoniansto second order

$$H_3 = \frac{p_x^2 + p_y^2}{2(1+\delta)} - \frac{q}{p_0} A_s + \frac{1}{2}(h_B\, x)^2 - h_B\, x\, \delta + O(3)$$

where we have subtracted the dipole field fromA_s

$$\frac{q}{p_0} A_s \to \frac{q}{p_0} A_s + \frac{1}{2}(1 + h_B\, x)$$

with h_{ref} has been chosen equal to h_B and by assuming the curvature h_{ref} to be small ("large ring"). Using the multipole expansion

$$(B_y + i\,B_x) = (B\,\rho_{ref}) \sum_{n=1}^{N} (i\,a_n + b_n)(x + i\,y)^{n-1}$$

neglecting end-fields. It is clear that

$$\frac{q}{p} = -\frac{1}{(B\,\rho_B)}$$

which is known as the "magnetic rigidity". It follows that

$$\boxed{h_B = b_1}$$

With a suitable choice of gauge we find the corresponding vectorpotential to be

$$\frac{q}{p_0} A_x = 0 ,$$

$$\frac{q}{p_0} A_y = 0 ,$$

$$\frac{q}{p_0} A_s = -\mathrm{Re} \sum_{n=1}^{N} \frac{1}{n} (i\, a_n + b_n)(x + i\, y)^n$$

where a_n and b_n are the skew- and normal multipolecoefficients.

Hamiliton's equations are

$$\begin{aligned}
x' &= \frac{\partial H}{\partial p_x} = \frac{p_x}{1+\delta} + O(2) , \\
p_x' &= -\frac{\partial H}{\partial x} = \frac{q}{p_0} B_y + h_B\, \delta - h_B^2\, x + O(2) , \\
y' &= \frac{\partial H}{\partial p_y} = \frac{p_y}{1+\delta} + O(2) , \\
p_y' &= -\frac{\partial H}{\partial y} = -\frac{q}{p_0} B_x + O(2) , \\
-cT' &= \frac{\partial H}{\partial \delta} = h_B\, x + O(2)
\end{aligned}$$

4 × 5 Matrix Formalism

Matrix style codes computes the solutions of Hamilton'sequations as Taylor expansion around a reference curve x_{ref}

$$x_j^f = \sum_k M_{jk}\, x_k^i + \sum_{kl} T_{jkl}\, x_k^i\, x_l^i + \cdots$$

where $\bar{x} = (x, p_x, y, p_y, \delta)$ and M_{jk} is the Jacobian

$$M = \left.\frac{\partial(x^f, p_x^f, y^f, p_y^f, \delta)}{\partial(x^i, p_x^i, y^i, p_y^i, \delta)}\right|_{\bar{x}_i = \bar{x}_{ref}}$$

In other words, M is the 4 × 5 linear transport matrixacting on the phase space vector $\bar{x} = (x, p_x, y, p_y, \delta)$. It is customary to choose the closed orbit as the referencecurve for circular accelerators. Note that, x_k^i is a contraviant vector, $x_k^i\, x_l^i$ a contravariant second rank tensor etc.

If only linear terms are kept

$$\bar{x}^f = M\,\bar{x}^i + O(\bar{x}^2)$$

The motion is symplectic since the equation of motions are derivedfrom a Hamiltonian . It follows that

$$\det M = 1$$

Since the higher order terms violates the symplectic condition,thin kicks are used for the higher order multipoles. The magnetmodel used for 4 × 5 matrix style calculations isshown in Fig. 1. Each magnet is broken up into two halves, representedby a linear matrix, and a thin kick at the center, containingthe higher order multipoles.

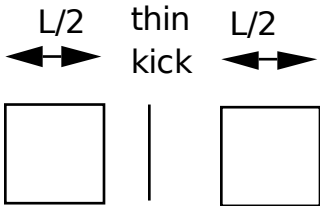

Fig. 1: The Magnet Model

Extended 4 × 5 Matrix Formalism Including ThinDipole Kicks

The 4 × 5 matrix formalism

$$\overline{x}^f = M\,\overline{x}^i$$

can be extended to include dipole kicks

$$\boxed{\overline{x}^f = \begin{pmatrix} 0 \\ -b_1 L \\ 0 \\ a_1 L \\ 0 \end{pmatrix} + M\,\overline{x}^i}$$

by superposition. The column vector describing the dipole kickcan therefore be implemented by adding this as a 6:th column anda 6:th row with (0, 0, 0, 0, 0, 0, 1) to the matrix. The normalrule for matrix multiplication is then applied and it is possibleto concatenate all linear elements, including dipole kicks andmis-alignments.

The Combined Function Sector Bend

In the focusing plane []

$$\begin{pmatrix} \cos\phi & \frac{1}{\sqrt{|K|}}\sin\phi & \frac{h_B}{|K|}(1-\cos\phi) \\ -\sqrt{|K|}\sin\phi & \cos\phi & \frac{h_B}{\sqrt{|K|}}\sin\phi \\ 0 & 0 & 1 \end{pmatrix}$$

and the defocusing plane

$$\begin{pmatrix} \cosh\phi & \frac{1}{\sqrt{|K|}}\sinh\phi & \frac{h_B}{|K|}(\cosh\phi-1) \\ \sqrt{|K|}\sinh\phi & \cosh\phi & \frac{h_B}{\sqrt{|K|}}\sinh\phi \\ 0 & 0 & 1 \end{pmatrix}$$

where

$$\phi \equiv L\sqrt{|K|}, \quad K \equiv \begin{cases} b_2 + h_B^2, & \text{horizontal plane} \\ b_2, & \text{vertical plane} \end{cases}$$

Edge Focusing

Leading order edge focusing is described by

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ h_B\tan(\psi) & 1 & 0 & 0 & 0 \\ 0 & 0 & -h_B\tan(\psi-\psi_c) & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

where ψ is the edge angle and ψ_c the leading order correction for a finite magnet gap, givenby

$$\psi_c = K_1\, h_B\, g\, \frac{1+\sin^2\psi}{\cos\psi}\,(1 - K_1\, K_2\, h_B\, g\tan\psi)$$

where g is the total magnet gap, $K_1 = 0.5$and $K_2 = 0$. Note that this implementationdoes not give the correct momentum dependence.

The Undulator

In the horizontal plane []

$$\begin{pmatrix} \cos\phi & \frac{1}{\sqrt{|K|}}\sin\phi & \frac{h_B}{|K|}(1-\cos\phi) \\ -\sqrt{|K|}\sin\phi & \cos\phi & \frac{h_B}{\sqrt{|K|}}\sin\phi \\ 0 & 0 & 1 \end{pmatrix}$$

and the vertical plane

$$\begin{pmatrix} \cosh\phi & \frac{1}{\sqrt{|K|}}\sinh\phi & \frac{h_B}{|K|}(\cosh\phi-1) \\ \sqrt{|K|}\sinh\phi & \cosh\phi & \frac{h_B}{\sqrt{|K|}}\sinh\phi \\ 0 & 0 & 1 \end{pmatrix}$$

The Thin Lens Approximation

Non-linear multipoles are modelled by thin kicks taking thelimit

$$L \to 0\,, \quad kL = \text{const}$$

where kL is the integrated strength. The kick is obtained by integratingHamilton's equations using delta functions for the multipolesand replacing the strength by integrated strength. We find

$$\boxed{\begin{aligned} p_x^f &= p_x^i - L\left(\frac{q}{p_0}B_y - h_B\,\delta + h_B^2\,x^i\right), \\ p_y^f &= p_y^i + \frac{q\,L}{p_0}B_x\,, \\ cT^f &= cT^i + h_B\,L\,x^i \end{aligned}}$$

assumingh_B to be small, where L is the length of the element. It is clearthat this model is symplectic. The corresponding linear matrixis given by

$$\left.\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -\frac{q\,L}{p_0}\frac{\check{Z}B_y}{\check{Z}x} - L\,h_B^2 & 1 & -\frac{q\,L}{p_0}\frac{\check{Z}B_y}{\check{Z}y} & 0 & L\,h_B \\ 0 & 0 & 1 & 0 & 0 \\ \frac{q\,L}{p_0}\frac{\check{Z}B_x}{\check{Z}x} & 0 & \frac{q\,L}{p_0}\frac{\check{Z}B_x}{\check{Z}y} & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}\right|_{\overline{x}=\overline{x}_{ref}}$$

where the field derivatives are computed from the multipole expansion.

The Cavity Model

If we neglect radial fields in the cavity it can be representedby a thin longitudinal kick

$$\delta^f = \delta^i - \frac{q\,\hat{V}_{RF}}{E_0}\sin\left(\frac{2\,\pi\,f_{RF}}{c}\,c\,T\right)$$

where E_0 is the beam energy,$\hat{V}_{RF}$ the cavity voltage and f_{RF} the RF frequency.Note that cT is the deviation of pathlength relative to a referenceparticle. To obtain absolute pathlength. the length of each magnetis added to the relative pathlength cT for each element and, atthe cavity, we subtract

$$c\,T^f = c\,T^i - \frac{h\,c}{f_{RF}}$$

where h is the harmonic number, to avoid numerical overflow forcT.

The Symplectic Integrator

It is possible to extend the 4×5 matrixformalism to the 6×6 case, as well as include higherorder effects, by using a (non-symplectic), e.g. second ordermatrix formalism [Brown]. However, this leads to a rather cumbersomeformulation. The elegant way, which also has the advantage ofbeing exact in the transverse coordinates, is to use a symplecticintegrator []. The importance of symplectic tracking for the studylong term stability is obvious.

The Hamiltonian is separated into two exactly solvable parts

$$\boxed{H_1 = H_4 + H_5}$$

where, neglecting fringe fields

$$\boxed{H_4 = -\,(1 + h_{ref}\,x)\,\sqrt{(1+\delta)^2 - p_x^2 - p_y^2} + \delta\,, \quad H_5 = -\,(1 + h_{ref}\,x)\,\frac{q}{p_0}\,A_s}$$

For efficiency we will use the expanded Hamiltonian

$$\boxed{H_4 = \frac{p_x^2 + p_y^2}{2\,(1+\delta)} + O(3)\,, \quad H_5 = -\,\frac{q}{p_0}\,A_s + \frac{1}{2}\,(h_B\,x)^2 - h_B\,x\,\delta + O(3)}$$

The map generated by H_1 is approximatedby a symplectic integrator. A 2:nd order integrator is given by[Ruth, Forest]

$$\boxed{\exp(:-L\,H_1:) = \exp\left(:-\frac{L}{2}\,H_4:\right)\exp(:-L\,H_5:)\exp\left(:-\frac{L}{2}\,H_4:\right) + O(L^3)}$$

Since H_4 is the Hamiltonian for a driftand H_5 corresponds to a thin kick, see Fig.2.

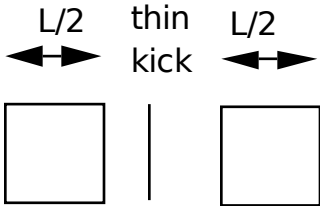

Fig. 2: A 2:nd order symplectic integrator.

Given a symmetric integrator of order 2n, $S_{2n}(L)$, a (2n + 2):th order integrator is obtained by [Yoshida]

$$S_{2n+2}(L) = S_{2n}(z_1 L)\, S_{2n}(z_0 L)\, S_{2n}(z_1 L) + O\left(L^{2n+3}\right)$$

where

$$z_0 = -\frac{2^{1/(2n+1)}}{2 - 2^{1/(2n+1)}}, \quad z_1 = \frac{1}{2 - 2^{1/(2n+1)}}$$

In particular, a 4:th order integrator is therefore given by

$$\begin{aligned} \exp(:-L H_1:) = {} & \exp(:-c_1 L H_4:) \exp(:-d_1 L H_5:) \exp(:-c_2 L H_4:) \exp(:-d_2 L H_5:) \\ & + \exp(:-c_2 L H_4:) \exp(:-d_1 L H_5:) \exp(:-c_1 L H_4:) + O\left(L^5\right) \end{aligned}$$

where

$$c_1 = \frac{1}{2\left(2 - 2^{1/3}\right)}, \quad c_2 = \frac{1 - 2^{1/3}}{2\left(2 - 2^{1/3}\right)}$$

$$d_1 = \frac{1}{2 - 2^{1/3}}, \quad d_2 = -\frac{2^{1/3}}{2 - 2^{1/3}}$$

see Fig. 3.

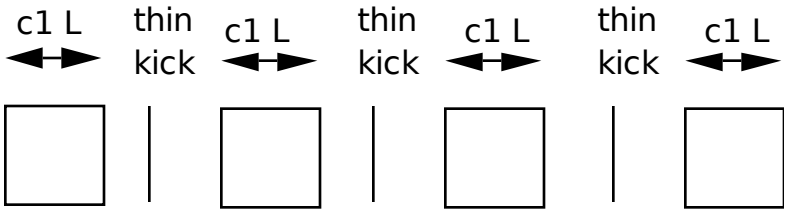

Fig. 3: A 4:th order symplectic integrator.

Both integrators are implemented.

Implementation of torsion...

Mis-alignments are implemented by applying a Euclidian transformationat the entrance and exit of each magnet [Forest]. We first transformto the magnets local coordinate system

$$\text{prot}\left(\frac{\phi}{2}\right) \circ R\left(\theta_{des}\right)$$

where R(θ) is a rotation in 2 dimensions

$$\begin{aligned} x &\leftarrow x \cos\left(\theta\right) + y \sin\left(\theta\right), \\ p_x &\leftarrow p_x \cos\left(\theta\right) + p_y \sin\left(\theta\right), \\ y &\leftarrow - x \sin\left(\theta\right) + y \cos\left(\theta\right), \\ p_y &\leftarrow - p_x \sin\left(\theta\right) + p_y \cos\left(\theta\right) \end{aligned}$$

with the design roll θ_{des} (e.g.a vertical bend is obtained by rotating a horizontal bend by 90°) and prot defined by

$$\begin{aligned} p_s &\leftarrow \sqrt{\left(1 + \delta\right)^2 - p_x^2 - p_y^2}\,, \\ x &\leftarrow \frac{x\, p_s}{p_s \cos\left(\phi/2\right) - p_x \sin\left(\phi/2\right)}, & p_x &\leftarrow p_s \sin\left(\frac{\phi}{2}\right) + p_x \cos\left(\frac{\phi}{2}\right), \\ y &\leftarrow y + \frac{x\, p_y \sin\left(\phi/2\right)}{p_s \cos\left(\phi/2\right) - p_x \sin\left(\phi/2\right)}, & p_y &\leftarrow p_y\,, \\ t &\leftarrow t + \frac{x\left(\frac{1}{\beta} + \delta\right) \sin\left(\phi/2\right)}{p_s \cos\left(\phi/2\right) - p_x \sin\left(\phi/2\right)}, & p_t &\leftarrow p_t \end{aligned}$$

where ϕ is the bend angle. If we expand and only keeplinear terms in the transverse coordinates as well as ϕ we find

$$\begin{aligned} x &\leftarrow x + O(2), & p_x &\leftarrow p_x + \sin\left(\frac{\phi}{2}\right) + O(2), \\ y &\leftarrow y + O(2), & p_y &\leftarrow p_y\,, \\ t &\leftarrow t + O(2), & p_t &\leftarrow p_t \end{aligned}$$

The Euclidian transformation consists of a translation T

$$\bar{x} \leftarrow T(\bar{x}) = \bar{x} - \Delta\bar{x}$$

followed by a rotation R with the total roll angle θ.The total misaligment has the following contributions

$$\Delta\bar{x} = \Delta\bar{x}_{sys} + \Delta\bar{x}_{rms}\, r$$

where r is a random number and similarly, the total roll angle

$$\theta = \theta_{des} + \Delta\theta_{sys} + \Delta\theta_{rms}\, r$$

where θ_{des} is a design tilt. Sincewe are now in the magnet's reference system we only have to applyprot(-ϕ/2) to transform back.

The multipole components have the following contributions

$$a_n = a_{n\,des} + a_{n\,sys} + a_{n\,rms}\, r$$
$$b_n = b_{n\,des} + b_{n\,sys} + b_{n\,rms}\, r$$

where $a_{n\,des}$ and $b_{n\,des}$ are the design multipole strengths.

The Euclidian Transformation

We summarize: at the entrance of a given magnet we applya Euclidian transformation

$$\text{prot}^{-1}\left(\frac{\phi}{2}\right) \circ R(\theta) \circ T(\Delta\,\bar{x}) \circ R^{-1} \circ (\theta_{des}) \circ \text{prot}\left(\frac{\phi}{2}\right) \circ R(\theta_{des})$$

The transformation

$$R^{-1} \circ (\theta_{des}) \circ \text{prot}(\phi/2) \circ R(\theta_{des})$$

is given by

$$x \leftarrow x + O(2), \quad p_x \leftarrow p_x + \sin\left(\frac{\phi}{2}\right)\cos(\theta_{des}) + O(2),$$
$$y \leftarrow y + O(2), \quad p_y \leftarrow p_y + \sin\left(\frac{\phi}{2}\right)\sin(\theta_{des}) + O(2),$$
$$t \leftarrow t + O(2), \quad p_t \leftarrow p_t$$

We then translate

$$\boxed{\begin{aligned} x &\leftarrow x - \Delta x\,, \\ y &\leftarrow y - \Delta y \end{aligned}}$$

rotate

$$\boxed{\begin{aligned} x &\leftarrow x\cos(\theta) + y\sin(\theta)\,, \\ p_x &\leftarrow p_x\cos(\theta) + p_y\sin(\theta)\,, \\ y &\leftarrow -x\sin(\theta) + y\cos(\theta)\,, \\ p_y &\leftarrow -p_x\sin(\theta) + p_y\cos(\theta) \end{aligned}}$$

and finally apply

$$\mathrm{prot}^{-1}(\phi/2) \circ R(\theta)$$

or

$$\boxed{\begin{aligned} &x \leftarrow x + O(2)\,, \quad p_x \leftarrow p_x - \sin\left(\frac{\phi}{2}\right) + O(2)\,, \\ &y \leftarrow y + O(2)\,, \quad p_y \leftarrow p_y + O(2)\,, \\ &t \leftarrow t + O(2)\,, \quad p_t \leftarrow p_t \end{aligned}}$$

We now integrate through the magnet. Similarly, at the exit weapply

$$R^{-1}(\theta_{des})\,\mathrm{prot}\left(\frac{\phi}{2}\right) R(\theta_{des})\, T^{-1}(\Delta\,\overline{x})\, R^{-1}(\theta)\;\; \mathrm{prot}^{-1}\left(\frac{\phi}{2}\right)$$

The corresponding matrix is

$$\begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 & 0 \\ 0 & \cos(\theta) & 0 & \sin(\theta) & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 & 0 \\ 0 & -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

since only the rotation contributes.

Note, that although the 4×5 matrix formalismcan be applied in the case of magnet errers this treatment isinconsistent, since the matrices are obtained by expanding aroundthe reference trajectory. In other words, only feed-down due tolinear terms, are

included for elements represented by matrices.This model should therefore, at most, be applied for linear latticedesign with no magnet errors. The use of a symplectic integratorand automatic differentiation (AD) allows for the implementationof a consistent model, since AD allows us to compute non-linearmaps around any reference curve and in particular, linear mapsaround the perturbed closed orbit.

The Closed Orbit Finder

For the 4×5 matrix formalism the linear one-turn-mapis computed by concatenating the linear transfer matrices.

In the case of the symplectic integrator, all the calculationsare performed using a package for truncated power series algebrato find the Taylor series expansion of the non-linear map M to arbitrary order. Given the purpose of this codeas well as for efficiency, we have linked to routines for linearpower series computing the linear map M. It is straightforward(more compact, efficient etc.) to write an independent code thatcomputes and analysis higher order maps by reading a machine filedescribing the lattice generated by this code.

The linear map M is calculated for a given reference trajectory.In the circular case the closed orbit is normally used. The closedorbit is different from the design orbit when misalign- and tilterrors are added for the magnets. In this case the closed orbithas to be found numerically.

For the one turn map we have

$$\boxed{\bar{x}_f = M\ \bar{x}_i}$$

The closed orbit at the starting point of the lattice is givenby the fixed point

$$M\ \bar{x}_{cod} = \bar{x}_{cod}$$

or

$$(M\ - I)\,\bar{x}_{cod} = 0$$

The fixed point is found numerically with Newton-Raphson's method[]

$$\boxed{f'(\bar{x}_k)\,(\bar{x}_{k+1} - \bar{x}_k) + f(\bar{x}_k) = 0}$$

where $f'(\bar{x}_k)$ the Jacobian. It follows

$$f(\bar{x}_i^k) = (M - I)\,\bar{x}_i^k = \bar{x}_f^k - \bar{x}_i^k, \quad f'(\bar{x}_i^k) = M - I$$

so that

$$\boxed{\bar{x}_i^{k+1} = \bar{x}_i^k - (M - I)^{-1}\,(\bar{x}_f^k - \bar{x}_i^k)}$$

Note that the linear one turn map M has to be calculated for eachiteration. The closed orbit at other points in the lattice arecomputed by tracking.

Linear Lattice Calculations

The linear equations of motion are obtained by expandingthe Hamiltonian to second order and assuming mid-plane symmetry

$$H_3 = \frac{p_x^2 + p_y^2}{2\left(1 + \delta\right)} + \frac{1}{2}\left[b_2(s) + h_B^2(s)\right] x^2 - \frac{1}{2} b_2(s)\, y^2 - h_B(s)\, x\, \delta + O(3)$$

with the solution

$$x = \sqrt{2\, J_x\, \beta_x(s)}\, \cos\left[\mu_x(s) + \varphi_x\right],$$

$$p_x = -\sqrt{\frac{2\, J_x}{\beta_x(s)}}\, \left\{\sin\left[\mu_x(s) + \varphi_x\right] + \alpha_x(s) \cos\left[\mu_x(s) + \varphi_x\right]\right\}$$

where

$$\alpha_x(s) \equiv -\frac{1}{2}\, \beta_x'(s)$$

The linear one-turn map M can in the 2×2 case be written

$$M = \begin{pmatrix} \cos\mu + \alpha \sin\mu & \beta \sin\mu \\ -\gamma \sin\mu & \cos\mu - \alpha \sin\mu \end{pmatrix}$$

where the phase advance $\mu(s)$ is given by

$$\mu(s) \equiv \int_{s0}^{s} \frac{d\tau}{\beta(\tau)}$$

and

$$\gamma \equiv \frac{1}{\beta}\left(1 + \alpha^2\right)$$

We apply the following canonical transformation A so that

$$A^{-1}\, M\, A = R(\mu) = \begin{pmatrix} \cos\mu & \sin\mu \\ -\sin\mu & \cos\mu \end{pmatrix}$$

where $R(\mu)$ is the 2-dimensional rotation matrix. We find

$$A = \begin{pmatrix} \frac{1}{\sqrt{\gamma}} & \frac{-\alpha}{\sqrt{\gamma}} \\ 0 & \sqrt{\gamma} \end{pmatrix}$$

If one imposes the normal definition of phase advance, the corresponding $A_{C\&S}$ is obtained from A by rotating withand angle of arctan(α)

$$A_{C\&S} = \begin{pmatrix} \frac{1}{\sqrt{\gamma}} & \frac{-\alpha}{\sqrt{\gamma}} \\ 0 & \sqrt{\gamma} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{1+\alpha^2}} & \frac{\alpha}{\sqrt{1+\alpha^2}} \\ \frac{-\alpha}{\sqrt{1+\alpha^2}} & \frac{1}{\sqrt{1+\alpha^2}} \end{pmatrix} = \begin{pmatrix} \sqrt{\beta} & 0 \\ \frac{-\alpha}{\sqrt{\beta}} & \frac{1}{\sqrt{\beta}} \end{pmatrix}$$

In the general case, the 4×5 one-turn-map is diagonalized and the corresponding A is concatenatedwith the transport matrices to compute the values of the latticefunctions after each element in the lattice. Linear coupling istherefore autamatically taken into account

The one-turn matrix has the form

$$M = \begin{pmatrix} & & & & 0 & n_{16} \\ & N & & & 0 & n_{26} \\ & & & & 0 & n_{36} \\ & & & & 0 & n_{46} \\ n_{51} & n_{52} & n_{53} & n_{54} & 1 & n_{56} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

It follows that the δ-dependent fix point is given by

$$\Delta \bar{x}_{cod} = \bar{\eta}\,\delta = N\,\bar{\eta}\,\delta + \bar{n}\,\delta$$

so that

$$\bar{\eta} = (I - N)^{-1}\,\bar{n}$$

where $\bar{\eta} = (\eta_x, \eta'_x, \eta_y, \eta'_y)$ is the linear dispersion. and $\bar{n} = (n_{16}, n_{26}, n_{36}, n_{46})$. The translation to this point in phase space can be done bythe translation operator

$$T = e^{:\overline{\Delta x}\cdot\ \overline{x}:}$$

where

$$:\overline{\Delta x}\cdot\ \overline{x}: = \frac{1}{2}\sum_{i\,j}\Delta x_i\ J_{i\,j}\ x_j$$

and $J_{i\,j}$ is the symplectic form

$$\overline{J} = \begin{pmatrix} \overline{0} & \overline{1} \\ -\overline{1} & \overline{0} \end{pmatrix}$$

Applying the corresponding canonical transformation B

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & \eta_x \\ 0 & 1 & 0 & 0 & 0 & \eta'_x \\ 0 & 0 & 1 & 0 & 0 & \eta_y \\ 0 & 0 & 0 & 1 & 0 & \eta'_y \\ -\eta'_x\ \eta_x & -\eta'_y\ \eta_y & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

and A as before we find

$$A^{-1}\ B^{-1}\ M\ B\ A = \begin{pmatrix} \cos\mu_x & \sin\mu_x & 0 & 0 & 0 & 0 \\ -\sin\mu_x & \cos\mu_x & 0 & 0 & 0 & 0 \\ 0 & 0 & \cos\mu_y & \sin\mu_y & 0 & 0 \\ 0 & 0 & -\sin\mu_y & \cos\mu_y & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & C\ \alpha_c \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

where α_c is the momentum compaction

$$\alpha_c \equiv \frac{1}{C}\frac{d\,(c\,T)}{d\,\delta}$$

and C the circumference. The longitudinal chromaticity η_δ is defined by

$$\eta_\delta \equiv \frac{1}{\omega}\frac{d\,\omega}{d\,\delta} = \frac{1}{\gamma_t^2} - \alpha_c = \frac{E_0^2 - \alpha_c\,E^2}{E^2}$$

and we have for the linearized eqution of motion

$$\boxed{\delta^f = \delta^i + \frac{q\,\hat{V}}{E_0}\sin\left\{\frac{\omega_{RF}}{c}\left[c\,T_0 + c\,T + T_i + C\,\alpha_c\,\delta + \overline{n}^T\cdot\ \overline{x}\right]\right\}}$$

For reference purposes we present the corresponding equation ofmotion using angle variables

$$\ddot{\phi} + \frac{\Omega^2}{\cos \phi_s} \left(\sin \phi - \sin \phi_s \right) = 0$$

where

$$\Omega = \sqrt{\frac{\omega_{RF} \, \alpha_c \cos \phi_s}{T_0} \frac{q \, \hat{V}_{RF}}{E_0}}$$

and

$$\phi = \frac{\omega_{RF}}{c} c \, T \, , \quad \dot{\phi} = \omega_{RF} \, \alpha_c \, \delta$$

Calculation of Tune for a General 4×4 Symplectic Matrix

The characteristic polynomial P(λ) of an arbitrarysymplectic matrix is given by [Forest]

$$P(\lambda) = \det(M - \lambda I) = (\lambda - \lambda_0)\left(\lambda - \frac{1}{\lambda_0}\right)(\lambda - \lambda_1)\left(\lambda - \frac{1}{\lambda_1}\right)$$

It follows that

$$P(1) = (2 - x)(2 - y), \quad P(-1) = (2 + x)(2 + y)$$

where

$$x = \lambda_0 + \frac{1}{\lambda_0} = 2 \cos(2\pi\nu_x)$$

and similarly for y. Eliminating y

$$x^2 + 4\,b\,x + 4\,c = 0$$

where

$$b = \frac{P(1) - P(-1)}{16}, \quad c = \frac{P(1) + P(-1)}{8} - 1$$

Solving for x

$$x = -2\left(b \pm \sqrt{b^2 - c}\right),$$

so that

$$\boxed{\nu_{x,y} = \frac{1}{2\pi} \cos^{-1}\left(\frac{x}{2}\right)}$$

The right quadrant ($0 \rightarrow 2\pi$) is determined from inspection of the map M.

Chromatic effects using the matrix formalism, arecalculated by numerical differentiation. In particular, the closedorbit is calculated for the on- as well as the off-momentum togetherwith the one turn map.

Synchrotron Radiation

The classical radiation from an accelerated relativisticelectron is given by [Sands p.98]

$$\boxed{\frac{d\,E}{d\,(c\,t)} = \frac{q^2\,c^2\,C_\gamma}{2\,\pi}\,E^2\,(\overline{B}_\perp)^2}$$

where

$$C_\gamma = \frac{4\,\pi}{3}\,\frac{r_e}{(m_e\,c^2)^3} = 8.846269 \times 10^{-5}\ \mathrm{m\ GeV^{-3}}$$

Since

$$\frac{d\,E}{d(c\,t)} = -\,p_0\,\frac{d\,p_t}{d\,t}$$

It follows

$$\boxed{\frac{d\,p_t}{d\,(c\,t)} = -\,\frac{c\,C_\gamma}{2\,\pi}\;p_0\,E_0^2\left(1 - \frac{p_0\,c}{E_0}\,p_t\right)^2\left(\frac{\overline{B}_\perp}{B\,\rho}\right)^2}$$

If we take the ultra-relativistic limit

$$\boxed{p_t \to -\,\delta\,, \quad p_0\,c \to E_0 \quad \text{when} \quad \beta \to 1}$$

we find

$$\boxed{\frac{d\,\delta}{d\,(c\,t)} = -\,\frac{C_\gamma\,E_0^3}{2\,\pi}\,(1+\delta)^2\left(\frac{\overline{B_\perp}}{B\,\rho}\right)^2\,, \quad \beta \to 1}$$

The transverse field is computed from

$$\overline{B}_\perp = \overline{B} \times \hat{e}_s$$

$$\boxed{r' \equiv \frac{d\,r}{d\,s} = \sqrt{(1 + h_B\,x)^2 + x'^2 + y'^2}}$$

$$\hat{e}_x = \frac{x'}{|r'|}, \quad \hat{e}_y = \frac{y'}{|r'|}, \quad \hat{e}_s = \frac{r'}{|r'|}$$

Since x' and y' are conserved [Sands p. 104] it follows fromHamilton's equations

$$\boxed{\begin{aligned} x' &= \frac{\partial H}{\partial p_x} = \frac{p_x}{1+\delta} + O(2), \\ y' &= \frac{\partial H}{\partial p_y} = \frac{p_y}{1+\delta} + O(2) \end{aligned}}$$

that

$$\boxed{\begin{aligned} p_x^f &= \frac{\left(1+\delta^f\right)}{1+\delta^i}\, p_x^i, \\ p_y^f &= \frac{\left(1+\delta^f\right)}{1+\delta^i}\, p_y^i \end{aligned}}$$

The Local Bump Method

Closed orbit correction with local bump method []. Localbump implies

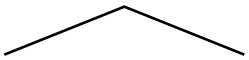

Fig. : Local bump

$$\theta_{x1} = \text{free parameter},$$

$$\theta_{x2} = -\sqrt{\frac{\beta_{x1}}{\beta_{x2}}}\,\frac{\sin(\mu_{x3} - \mu_{x1})}{\sin(\mu_{x3} - \mu_{x2})}\,\theta_{x1},$$

$$\theta_{x3} = -\sqrt{\frac{\beta_{x1}}{\beta_{x3}}}\,\frac{\sin(\mu_{x2} - \mu_{x1})}{\sin(\mu_{x3} - \mu_{x2})}\,\theta_{x1}$$

Least-squre minimization of the rms orbit

$$x_{rms}^2 = \sum_i \theta_{x1}\left(x_i + \sqrt{\beta_{x1}\,\beta_{xi}(s)}\,\sin(\mu_{xi} - \mu_{x1})\right)^2$$

gives

$$\theta_{x1} = -\frac{\sum_i x_i \sqrt{\beta_{x1}\,\beta_{xi}(s)}\,\sin(\mu_{xi} - \mu_{x1})}{\sum_i \left[\sqrt{\beta_{x1}\,\beta_{xi}(s)}\,\sin(\mu_{xi} - \mu_{x1})\right]^2}$$

In the linear approximation the new orbit is given by

$$x(s) = \begin{cases} -\sqrt{\beta_{x1}\,\beta_x(s)}\,\sin[\mu_x(s) - \mu_{x1}]\,\theta_{x1}, & s_1 \le s \le s_2 \\ -\sqrt{\beta_{x1}\,\beta_x(s)}\,\sin[\mu_x(s) - \mu_{x1}]\,\theta_{x1} + \sqrt{\beta_{x2}\,\beta_x(s)}\,\sin[\mu_x(s) - \mu_{x2}]\,\theta_{x2}, & s_2 \le s \le s_3 \end{cases}$$

Limited corrector strength is implented by successively scalingθ_1, θ_2, and θ_3 until reaching valuesthat are within limits.

The correlation matrix is given by

$$C_{ij} = \frac{\sqrt{\beta_i \, \beta_j}}{2 \sin(\pi \nu)} \cos\left[\pi \nu - |\mu_i - \mu_j|\right] + \eta_i \, \eta_j \, \delta$$

where the last term only contributes in the case of a cavity.We attempt to solve the following equation

$$C \, \bar{\theta}_x + \bar{x} = 0$$

It can be shown that

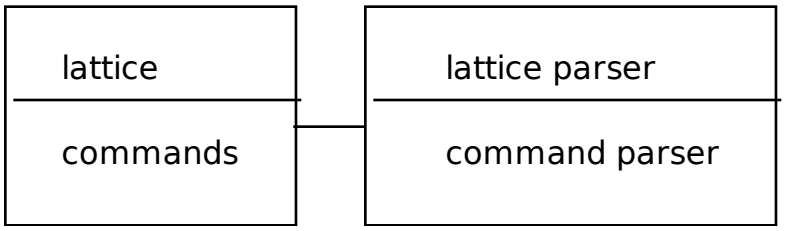

Fig. : Traditional tracking code structure

Fig. : TRACY program structure

Fig. : Pascal-S compiler/interpreter system

Tracy is using the following files

input file	*.inp
output file	*.out
lattice file	*.lat
lattice output file	*.lax

Omissions from Pascal-S

Pascal-S [] is a strict subset of Pascal []. It does notsupport the following

Data types	enumerated, subrangeand pointer
Data structures	variant record, packed, set and file
Statements	with and goto
Input/Output	put and get

Since the lack of text files is non-acceptable for our purpose,it has been added with some constraints for easy implementation.

There are four basic types of elements: drift, multipole,cavity and marker.

procedure **getelem**(i : integer; var cellrec : celltype);

Get cell record number i from internal data structures toPascal-S stack

input: i element number
output: cellrec element information
example: getcell(globval.Cell_nLoc, cell);

procedure **putelem**(i : integer; var cellrec : celltype);

Put cell record number i from Pascal-S stack to internaldata structures

input: cellrec element information
output: i element number
example: putcell(0, cell);

procedure **getglobv_**(globval);

procedure **putglobv_**(globval);

trace

break

text files and include files. There are essentially fivedifferent kinds of standard procedures and functions added toPascal-S:

a) string manipulations
b) some additional standard mathematical functions
c) matrix manipulations
d) graphics routines (strict subset of GKS)
e) accelerator physics routines

They are listed in the following.

String library

Data Structures

```
const  strlmax = 80;

type   strbuf    = packed array [1..strlmax] of char;
       tstring   = record
            len : integer;
            str : strbuf;
          end;
```

function **strlen_**(var str : tstring) : integer;

Get length of string:

input: str
output: strlen_
example: len := strlen_(str);

procedure **getstr**(var str : tstring;
vstr : packed array [low..high : integer]of char);

Get string:

input: vstr
output: str

example: getstr(str, 'This is a string');

procedure **copystr**(var outstr, instr : tstring);

Copy string:

input: instr
output: outstr
example: copystr(str2, str1);

procedure **concat**(var str2 : tstring;
str1 : packed array [low..high : integer]of char);

Concatenate strings:

input: str1
output: str2
example: concat(str, ' add this to string');

procedure **getint**(var str : tstring; i, blanks : integer);

Write integer into string:

input: i number to be written into str
blanks field width
output: str
example: getint(str, 1234, 10);
corresponds to: write(str, 1234:10);

procedure **getreal**(var str : tstring; x : double; blanks,ndec : integer);

Write real into string:

input: x number to be written into str
blanks field width
ndec number of decimals
output: str
example: getreal(str, 3.1415, 10, 5);
corresponds to: write(str, 3.1415:10:5);

procedure **getreale**(var str : tstring; x : double;blanks, ndec : integer);

Write real into string, exponential form:

input: x number to be written into str
blanks field with
ndec number of decimals
output: str
example: getreale(str, i, 10, 5);
corresponds to: write(str, x:10+5);

function **strind**(var object: tstring;
pattern : packed array [low..high : integer]of char)
: integer;

Pattern matching:

input: object
pattern
output: strind 0, if pattern not found, location,if pattern found
example: pos := strind(str, 'where');

procedure **writestr**(var str : tstring; blanks : integer;var outf : text);

Write string to file:

input: str
blanks field width
output: outf
example: writestr(str, 80, outfile);

Data Structures

```
const matdim = 6;

type   double   = real8;
       vector   = array [1..matdim] of double;
       matrix   = array [1..matdim] of vector;

var    pi         : double;
       rseed0, rseed     : integer;
       normcut_          : double;
```

Matdim is the maximum allowed matrix dimension.

FUNCTION **dble**(x : real) : double;

Convert from single to double precision

input: x
output: dble
example: z := dble(1.1);

FUNCTION **sngl**(x : double) : real;

Convert from double to singe precision:

input: x
output: sngl
example: z := sngl(2.1);

FUNCTION **min_**(x1, x2 : double) : double;

Get min value of x1 and x2

input: x1, x2
output: min_
example: xmin := min_(x1, x2);

FUNCTION **max_**(x1, x2 : double) : double;

Get max value of x1 and x2

input: x1, x2
output: max_
example: xmax := max_(x1, x2);

FUNCTION **power**(x, y : double) : double;

Evaluates x^y

input: x, y
output: power
example: z := power(2.0, 4);

FUNCTION **tan_**(x : double) : double;

Evaluates tan(x)

input: x
output: tan_
example:

FUNCTION **cosh_**(x : double) : double;

Evaluates cosh(x)

input: x
output: cosh
example:

FUNCTION **sinh_**(x : double) : double;

Evaluetes sinh(x)

input: x
output: sinh_
example:

FUNCTION **tanh_**(x : double) : double;

Evaluates tanh(x)

PROCEDURE **iniranf**(i : integer);

Initialize random number generator

input: i
output:
example:

PROCEDURE **newseed**;

Get a new seed for random number generator

input:
output:
example:

FUNCTION **ranf** : double;

Random number generator with rectangular distribution

input:
output: ranf
example:

PROCEDURE **setrancut**(cut : double);

Set cut for normally distributed random number generator

input: cut
output:
example:

FUNCTION **normranf** : double;

Random number generator with normal distribution

input:
output: normranf
example:

Conversion routines

FUNCTION **degtorad** (d : double) : double;

Degrees to radianer

input: d
output: degtorad
example:

FUNCTION **sign**(x : double) : integer;

Get sign of value

input: x
output: sign
example:

Function **GetAngle**(x, y : double) : double;

Get phi from x=cos(phi), y=sin(phi), -pi ≤ phi≤ pi

input: x, y
output: GetAngle
example:

Matrix routines

For the following routines n is the dimension of the vectorsand matrices.

PROCEDURE **UnitMat**(n : integer; VAR A : matrix);

Unit matrix: A ¬ I

input: A, n
output: A
example:

PROCEDURE **CopyVec**(n : integer; VAR u, v : vector);

Copy vector: v ¬ u;

input: u, n
output: v
example:

PROCEDURE **CopyMat**(n : integer; VAR A, B : matrix);

Copy matrix: B ¬ A

input: A, n
output: B
example:

PROCEDURE **AddVec**(n : integer; VAR a, b : vector);

Add vectors: b ¬ a + b

input: a, b, n
output: b
example:

PROCEDURE **SubVec**(n : integer; VAR u, v : vector);

Subtract vectors: u ¬ v - u

input: u, v, n
output: v
example:

PROCEDURE **AddMat**(n : integer; VAR A, B : matrix);

Add matrices: A ¬ A + B

input: A, B, n
output: B
example:

PROCEDURE **SubMat**(n : integer; VAR A, B : matrix);

Subtract matrices: B ¬ B - A

input: A, B, n
output: B
example:

PROCEDURE **LinTrans**(n : integer; VAR A : matrix; VARx : vector);

Linear transformation: x ¬ A * x

input: A, x, n
output: x
example:

PROCEDURE **MulLMat**(n : integer; VAR A, B : matrix);

Left matrix multiplication: B ¬ A * B

input: A, B, n
output: B
example:

PROCEDURE **MulRMat**(n : integer; VAR A, B : matrix);

Right matrix multiplication: A ¬ A * B

input: A, B, n
output: A
example:

FUNCTION **TrMat**(n : integer; VAR A : matrix) : double;

Trace: A ¬ Tr(A)

input: A, n
output: A
example:

PROCEDURE **TpMat**(n : integer; VAR A : matrix);

Transpose: $A \leftarrow A^T$

input: A, n
output: A
example:

FUNCTION **DetMat**(n : integer; VAR A : matrix) : double;

Determinant: A ¬ |A|

input: A, n
output: A
example:

function **InvMat**(n : integer; VAR A : matrix) : boolean;

Inverse: $A \leftarrow A^{-1}$

input: A, n
output: A
example:

procedure **prtmat**(n : integer; var A : matrix);

Print matrix on terminal

input: A, n
output:
example:

function **GetnKid**(Fnum1 : integer) : integer;

Get number of elements (kids) in for a given family

function **Elem_GetPos**(Fnum1, Knum1 : integer) : integer;

Get element number (1 - globval.cell_nloc)

procedure **Cell_SetdP**(dP : double);

input: dP
output:
example:

procedure **Cell_Pass**(i0, i1 : integer; var x : vector;var lastpos : integer);

Track particle from i0 to i1

input: i0 initial position
i1 final position
x initial conditions (x, px, y, py, delta, ctau)

output: x final conditions (x, px, y, py, delta,ctau)
lastpos last position (# i1 if particle is lost)

example: x[1] := x0; x[2] := px0; x[3] := y0; x[4] :=py0;
x[5] := delta; x[6] := 0.0;
Cell_Pass(0, globval.Cell_nLoc, x, lastpos);

procedure **Cell_Pass_M**(i0, i1 : integer; var xref :vector; var mat : matrix; var lastpos : integer);

Track matrix from i0 to i1 around ref. orbit

input: i0, i1
xref reference orbit
mat
output: mat
lastpos
example:

procedure **Cell_DApass**(i0, i1 : integer; var map :DAmap; var lastpos : integer);

Track matrix from i0 to i1 around ref. orbit, using DA

input: i0, i1
map
output: map
lastpos
example:

procedure **Cell_Concat**(dP : double);

Concatenate lattice for fast tracking

input: dP
output:
example:

procedure **Cell_fPass**(var x : vector; var lastpos :integer);

Fast tracking of particle using concatenated lattice

input: x
output: x
lastpos
example:

procedure **Cell_fPass_M**(var xref : vector; var mat: matrix; var lastpos : integer);

Fast tracking of matrix using concatenated lattice

input: xref
mat
output: mat
lastpos
example:

procedure **Cell_GetCOD**(imax : integer; eps, dP : double;var lastpos : integer);

Closed orbit finder

input: imax, eps, dP
output: laspos
example:

PROCEDURE **Cell_GetABGN**(var M : matrix; var alpha,beta, gamma,nu : vector2);

Get alpha, beta, gamma and nu from transport matrix

input: M
output: alpha, beta, gamma, nu
example:

procedure **Cell_MatTwiss**(i0, i1 : integer; var Ascr: matrix; chroma, ring : boolean);

Track A script from i0 to i1

input: i0, i1, chroma, ring
Ascr
output: Ascr
example:

procedure **Cell_DATwiss**(i0, i1 : integer; var Ascr: DAmap; chroma, ring : boolean);

Track A script from i0 to i1 using DA

input: i0, i1, chroma, ring
Ascr
output: Ascr
example:

procedure **Ring_Getchrom**(dP : double);

Get chromaticity

input: dP

output:
example:

procedure **Ring_GetTwiss**(chroma : boolean; dP : double);

Get Twiss parameters around lattice

input: chroma, dP
output:
example:

PROCEDURE **Ring_Fittune**(var nu : vector2; eps : double;var q : ivector2; dkL : double; imax : integer);

Fit tune

input: nu, eps, q, dkL, imax
output:
example:

PROCEDURE **Ring_Fitchrom**(var ksi : vector2; eps : double;var s : ivector2; dkpL : double; imax : integer);

Fit chromaticity

input: ksi, eps, s, dkpL, imax
output:
example:

PROCEDURE **Ring_FitDisp**(pos : integer; eta, eps : double;q : integer; dkL : double; imax : integer);

Fit dispersion

input: pos, eta, eps, q, dkL, imax
output:
example:

procedure **InitBUMP**(dnuhmin, dnuvmin : double);

Initialize orbit correction algorithm. It is necessary to call Ring_gettwiss before initbump can be called.

input: dnuhmin, dnuvmin
output:
example: Ring_gettwiss(false, 0.0);
InitBump(0.0, 0.0);

procedure **execbump**(MaxKick : double);

Do one iteration of orbit correction

input: MaxKick
output:
example:

Include Files

The following files are called include files and appear atthe input file level. They define generally useful specializedhigh level physics routines based on the more general low levelstandard procedures and functions.

physlib.inc

Data Structures

```
const  nueps   = 1d-6; nudk   = 0.001; nuimax   = 10;
       ksieps  = 1d-6; ksidkp = 0.01;  ksiimax  = 10;
       dispeps = 1d-4; dispdk = 0.2;   dispimax = 10;
```

Procedures and Functions

procedure **printglob**;

Print global values

input:
output:
example:
```
Ring_gettwiss(true, 0.0);
getglobv_(globval);
printglob;
```

procedure **printmat**(n : integer; var A : matrix; varoutf : text);

Print matrix to file

input: n, A, outf
output:
example:

procedure **printcellf**;

Print Twiss parameters

input:
output:

example:

procedure **Printcod**;

Print closed orbit

input:
output:
example:

procedure **getmean**(n : integer; var x : graphvect);

Remove average value from a set of data

input: n number of data
x data
output: x
example:

procedure **getcod**(dP : double; var lastpos : integer);

Get closed orbit

input: dP
output: lastpos
example:

procedure **TraceABN**(i0, i1 : integer; alpha, beta,eta, etap : Vector2);

Get alpha and beta from i0 to i1

input: i0, i1, alpha, beta, eta, etap
output:
example:

procedure **ttwiss**(alpha, beta, eta, etap : vector2;dP : double);

Get alpha and beta along lattice

input: alpha, beta, eta, etap, dP

output:
example:

PROCEDURE **FitTune**(qf, qd : integer; nux, nuy : double);

Fit tune

input: qf, qd, nux, nuy
output:
example:

PROCEDURE **FitChrom**(sf, sd : integer; ksix, ksiy :double);

Fit chromaticity

input: sf, sd, ksix, ksiy
output:
example:

PROCEDURE **FitDisp**(q, pos : integer; eta : double);

Fit dispersion

input: q, pos, eta
output:
example:

procedure **getfloqs**(var x : vector);

Transform to Floquet space

input: x
output: x
example:

procedure **track**(x, px, y, py, dp : double; nmax :integer;
var lastn, lastpos : integer; floq : boolean);

Track particle nmax turns around the closed orbti. Data is
stored in the file tracking.dat. Ring_Gettwiss must be

called first.

input: x, px, y, py, dp
nmax, floq
output: lastn, lastpos
example: Ring_gettwiss(true, delta);
track(x0, px0, y0, py0, delta,
nturn, lastn, lastpos, true);
if lastn <> nturn then writeln('Particle lost duringturn ', nturn:1, ' , at element ', lastpos:1);

procedure **getdynap**(var r0, dr0 : vector; dp, eps :double; napp : integer; var rapp : vector);

Get dynamical aperture

input: r0, dr0, dp, eps, napp
output: rapp
example:

procedure **gettrack**(var n : integer; var x, px, y,py : graphvect);

Get tracking data from file. Track must be called first.

input: n
output: x, px, y, py
example: Ring_gettwiss(true, delta);
track(x0, px0, y0, py0, delta,
nturn, lastn, lastpos, true);
if lastn <> n then writeln('Particle lost during turn', n:1, ' , at element ', lastpos:1);
gettrack(n, x, px, y, py);

procedure **getj**(n : integer; var x, px, y, py : graphvect);

Get linear invariant

input: n, x, px, y, py
output: x, y
example: gettrack(n, x, px, y, py);
getj(n, x, px, y, py);

procedure **getphi**(n : integer; var x, px, y, py : graphvect);

Get phase

input: n, x, px, y, py
output: x, y
example: gettrack(n, x, px, y, py);
getphi(n, x, px, y, py);

procedure **setdS**(Fnum : integer; dxrms, dyrms : double);

Set displacement errors

input: Fnum, dxrms, dyrms
output:
example:

procedure **setdT**(Fnum : integer; dTrms : double);

Set tilt errors

input: Fnum, dTrms
output:
example:

procedure **setdk**(Fnum, Order : integer; dksys,dkrms : double);

Set multipole errors

input: Fnum, Order, dksys, dkrms
output:
example:

procedure **plotfft**(wn, n : integer; var x : graphvect);

Plot DFT

input: wn, n, x
output:
example:

procedure **plotdynap**(r0, dp, eps : double; npoint,napp : integer);

Plot dynamical aperture

input: r0, dp, eps, npoint, napp
output:
example:

procedure **plotps**;

Plot phase space

input:
output:
example:

procedure plotj;

Plot linear invariant

input:
output:
example:

procedure **plotphi**;

Plot phase

input:
output:
example:

procedure **plotpos**(lastpos : integer);

Plot beam position

input: lastpos
output:
example:

procedure **plotcell**(symfac : integer);

Plot Twiss functions

input: symfac
output:
example:

procedure plotcorr;

Plot orbit corrector strengths

input:
output:
example:

procedure **plotcod**;

Plot closed orbit

input:
output:
example:

procedure **codcorrect**(bumpimax : integer; thetamax: double);

Closed orbit correction

input: bumpimax, thetamax
output:
example:

References

K. Jensen and N. Wirth, "Pascal User Manual and Report", (Springer-Verlag,1975)

N. Wirth, "Pascal-S: a subset and its implementation, in Pascal- The Language and Its Implementation", ed. D. W. Barron, pp.199-260, (Wiley, 1981)

R. E. Berry, "Programming Language Translation", (Ellis Horwood,1983)

M. Rees and D. Robson, "Practical Compiling with Pascal-S",(Addison-Wesley, 1988)

E. Forest

H. Nishimura