ETEAPOT Overview and Best Practices

John Talman

January 7, 2015

Abstract

- ETEAPOT (Electric Thin Element Accelerator Program for Optics and Tracking) clones TEAPOT/UAL (Unified Accelerator Libraries) functionality, but with electric rather than magnetic elements.
- Spin propagation, needed for EDM simulation, has been added. It "hitch hikes" on the orbit without influencing it.
- The powerful UAL architecture (adxf, sxf, apdf, ...) facilitates this.
- Each particle has its own "bend plane" in every lattice element.
- TEAPOT, based on the drift/kick/drift paradigm, has all "thin" elements (not including drifts) even including bends. All are described by general "multipole kicks".
- ETEAPOT treats bends faithfully as "thick". Drifts are thick in both treatments as the beam advances longitudinally in them. This requires bends to be treated quite differently than other multipoles in ETEAPOT.
- The code is in an "operational prototype" state at present. Moving to a "release" state will be straightforward.
- Implications for parallel processing.
- Best practices free for all.

3 Outline

4 Calculational Approach

1. Calculational Strategy
   - exact tracking in idealized lattice
   - orbit equation and BMT equation are solved exactly in the idealized electric bends and multipoles
   - solutions are joined smoothly
   - since solutions are exact, no artificial symplectification is required
   - spurious damping (or anti-damping) lifetime is greater than 10^6 turns
2. Physics Flow Chart
3. Computer Program Flow Chart
4. Computer Performance
   - circa (ca) 10^3 factor between computer time and real time
   - prospects for multiprocessing

5 Code Status (all in http://code.google.com/p/ual/source/list)

- An effective platform for modeling electric accelerators is needed for EDM (Electric Dipole Moment) experiments.
- ETEAPOT is such a platform, it has been through four years of on again, off again, critique/feedback, constituting something like one full ordinary development cycle. It is still in development, however.
- The
  - Transverse
  - Longitudinal
  - Spin

  phases can be viewed as subcycles.
- A best practices recommendation is to have all coding match the governing physics document precisely. Accompanying diagrams are essential. This has been largely achieved in an ongoing "symbolic/syntactic" (with testing!) approach.
- Code release ("E2") is anticipated in the near future. Actually, a "beta" release is anticipated.

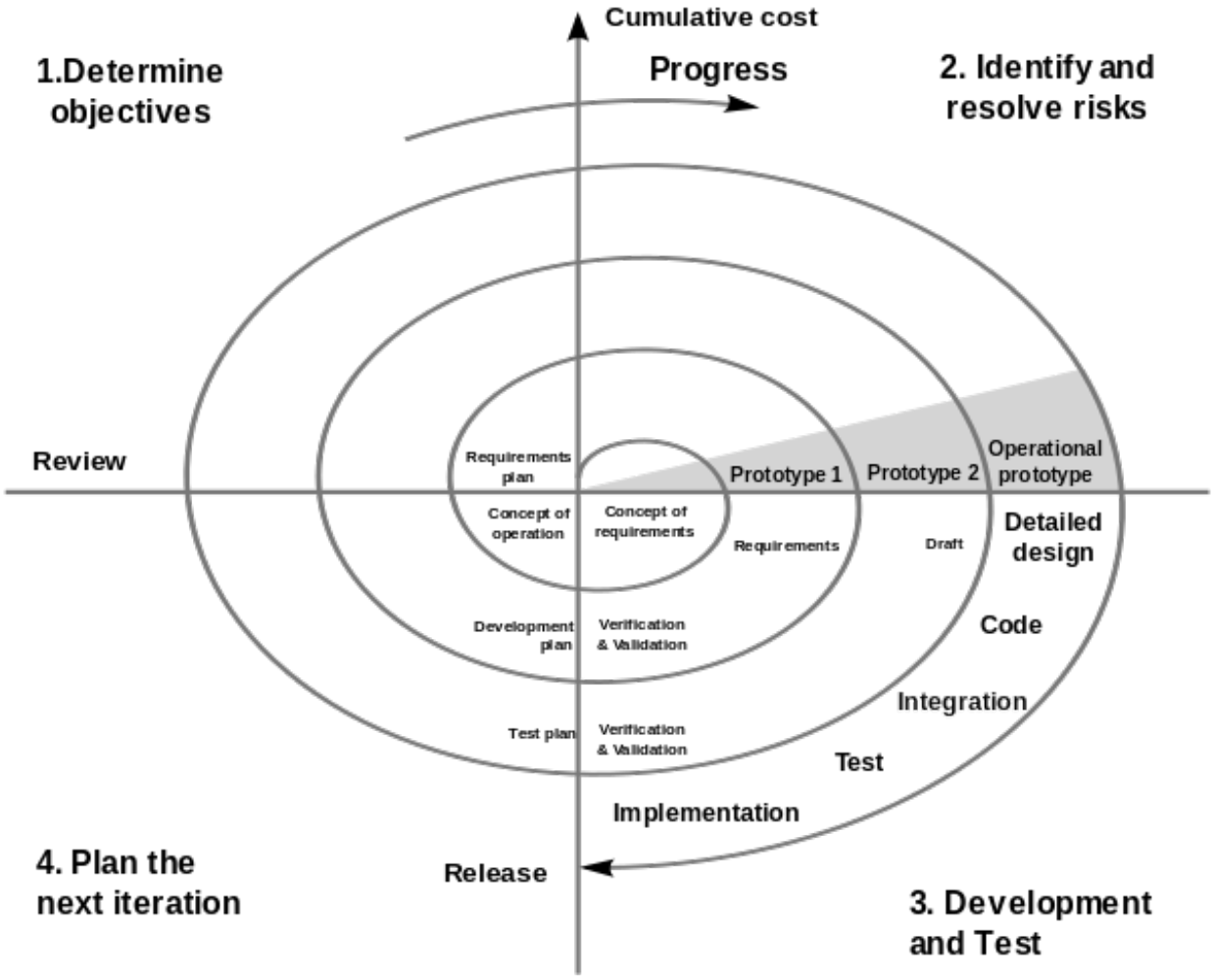

Figure: Software engineer's view of code project development. Somewhat artificially, "verification and validation" become "test plan", then "test" in the release cycle.

- ADXF, Accelerator Description eXchange Format:
  - XML based
  - Idealized lattice description with symbolic parameters.
  - Essential part of UAL lattice design.
- SXF, Standard eXchange Format:
  - Fully instantiated, purely numerical, lattice description.
  - Essential for UAL simulation.
- APDF, Accelerator Propagator Description Format:
  - Links algorithms with lattice elements.
  - Suports algorithm switching.

8 ETEAPOT2 Dynamical Reformulation of the Tracking Paradigm

- The traditional accelerator orbit formalism is geometric, modeled upon geometric optics; bends are through a given angle over a given arc length, quads are characterized by their focal lengths. Deviations from this paradigm are handled perturbatively e.g. by introducing dispersion. The constancy of velocity in magnetic rings makes this approach both natural and powerful. In this approach, the arc length, s, along the design orbit is the natural independent variable.
- The main distinction between electric and magnetic rings follows from the variation of particle speed in electric rings.
- To reflect this distinction, time, t, is taken to be the independent variable in ETEAPOT2. However, this distinction is more conceptual than it is numerical. This change can be made without seriously disrupting the UAL coordinatization. At the numerical level, it is only perturbations from the paradigm that are affected.
- For all elements treated (after slicing) as zero length, such as quadrupoles and rf cavities, no changes are necessary in this change of independent variables.

9 (Modified) Propagation Overview

- Drift (thick)
  - Used as part of the thin element model, typically.
  - Orbit: $\vec{x}_{output} = \vec{x}_{input} + \vec{v}_{input} t$, $\vec{p}_{output} = \vec{p}_{input}$
  - Spin: $\vec{s}_{out} = \vec{s}_{in}$
- Quad/Sext (thin$\equiv$ kick):
  - Neglect potential energy changes inside multipoles.
  - Orbit: (split if necessary) thin lens kicks e.g. $\frac{\Delta p_x}{p} = qx$. More detail below.
  - Spin: Closed form (near identity) 3D rotation matrix multiplication.
- Bend (thick): (expanded detail below)
  - Orbit: FF/Exact/(virtual)Kick/Exact/FF
  - Spin: BMT equation in bend plane is solved exactly.
- RF Cavity
  - Conceptually, time is the natural independent variable.

10 Categories of UAL Orbit Deviation

UAL tracking, like most accelerator tracking codes, takes the Poincare approach of tracking deviations from a nominal orbit.

For an ideal ring, the nominal orbit is taken to be the "design" orbit passing through the centers of all perfectly sited bends, drifts, and multipoles.

In any actual ring, survey positioning errors and element strength errors make it necessary to use a different definition of "nominal". A theorem due to Courant guarantees the existence of a unique central closed orbit for any magnetic accelerator. We assume (without proof) that the same theorem is applicable for electric rings and refer to this as the "deviant" closed orbit.

UAL code tracks deviations from this deviant closed orbit. This deviant closed orbit is designated as the "D-orbit".

The ETEAPOT2 code has not, as yet, allowed survey, electric, or magnetic errors. While this remains true, it is not wrong to interpret the "D" as standing for "design".

11 UAL Orbit Deviation Methodology I

Six independent orbital coordinates are tracked and kept in PAC::Position[0] through PAC::Position[5]. They are symbolized pacP[...] in the code. As explained above, these are deviations from the D-orbit, with it's Frenet-Serret basis vectors (three well defined directions, x, y, and z) which can be characterized as "Out" (X), "Up" (Y), and "Forward" (Z).

Thus pacP[0]= x. Like all PAC::Position[i], x is an evolving quantity, specific to the particular orbit/probe and elapsed time in that orbit.

The UAL architecture, with it's SXF lattice element description, discretizes the probe propagation so that $x \equiv x_E$, where E stands for the event of one of these discrete points.

As in systems engineering, probe evolution proceeds from input event to output event.

Element "splitting" complicates this evolution a little, but will not be detailed here. It basically logically separates an element into simple systems.

Thus the relevant propagation across/through a (possibly split) element involves two points, an input and an output.

Now pacP[1]=$x' \equiv \frac{dx}{dz} = \frac{dx}{dt}\frac{dt}{dz} = \frac{dx}{dt}/\frac{dz}{dt} = v_x/v_z = \gamma m v_x/\gamma m v_z = p_x/p_z \approx \frac{p_x}{p_D}$.

The element length is symbolized m_l in the code.

For drifts,

$x_{output} = x_{input} + v_{input\,X} * tof$

$y_{output} = y_{input} + v_{input\,Y} * tof$

Where tof is the probe time of flight through the element.

This is because, except for rigid parallel displacement, the relevant Frenet-Serret reference system is constant throughout the drift.

13 UAL Orbit Deviation Methodology II

The information on the actual time of flight is maintained as a deviation in pacP[4].

Thus this all appears in the code as

$$
\begin{aligned}
pacP[0] &= xoutput;\\
pacP[2] &= youtput;\\
pacP[4] &-= UAL :: clight * (tofD - tof);
\end{aligned}
$$

with $pacP[1]$, $pacP[3]$, and $pacP[5]$, all unchanged in a drift.

The explicit code for the drift propagation is in file

$UAL/codes/ETEAPOT2/src/ETEAPOT2/Integrator/driftMethods/propagate

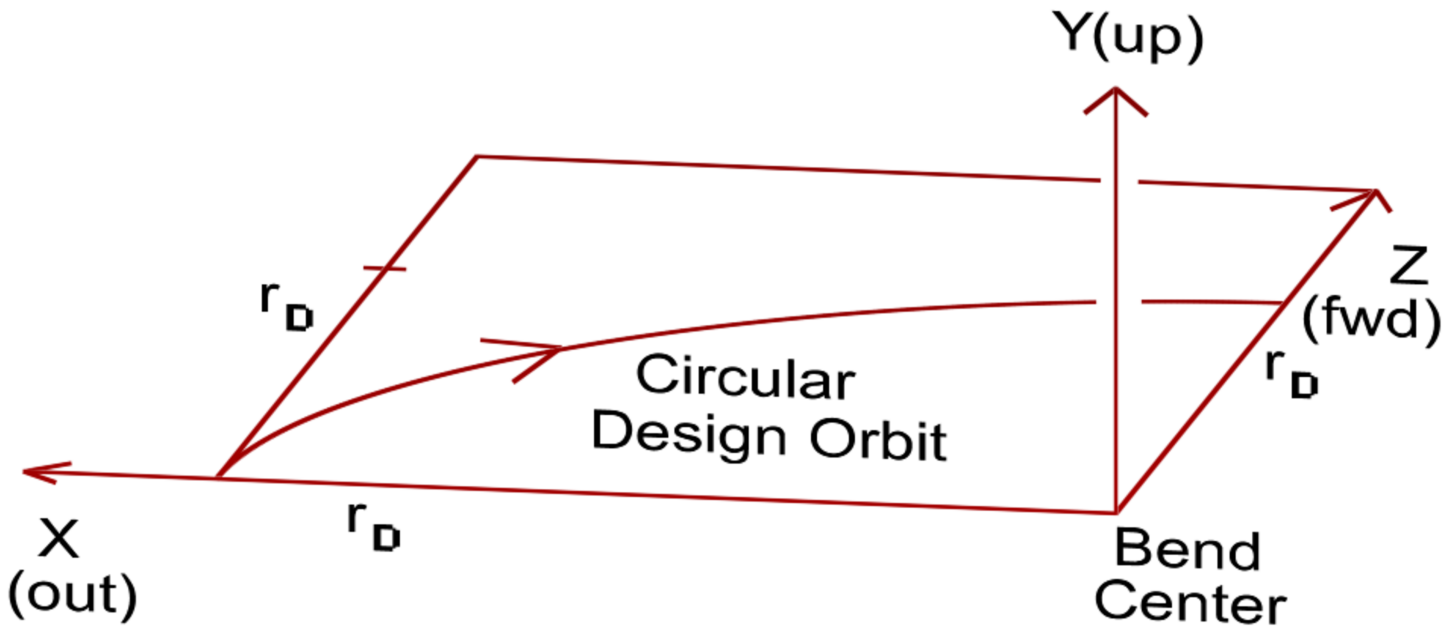

Figure: Design plane (Y = 0) orientation. A known, well defined, design/ideal particle reference orbit is central to the whole paradigm. Of course, this is not just spatial. Actual particle deviations from the reference particle are what are maintained (tracked). In the sxf files used in ETEAPOT, the design orbit in bend elements is a circular arc.

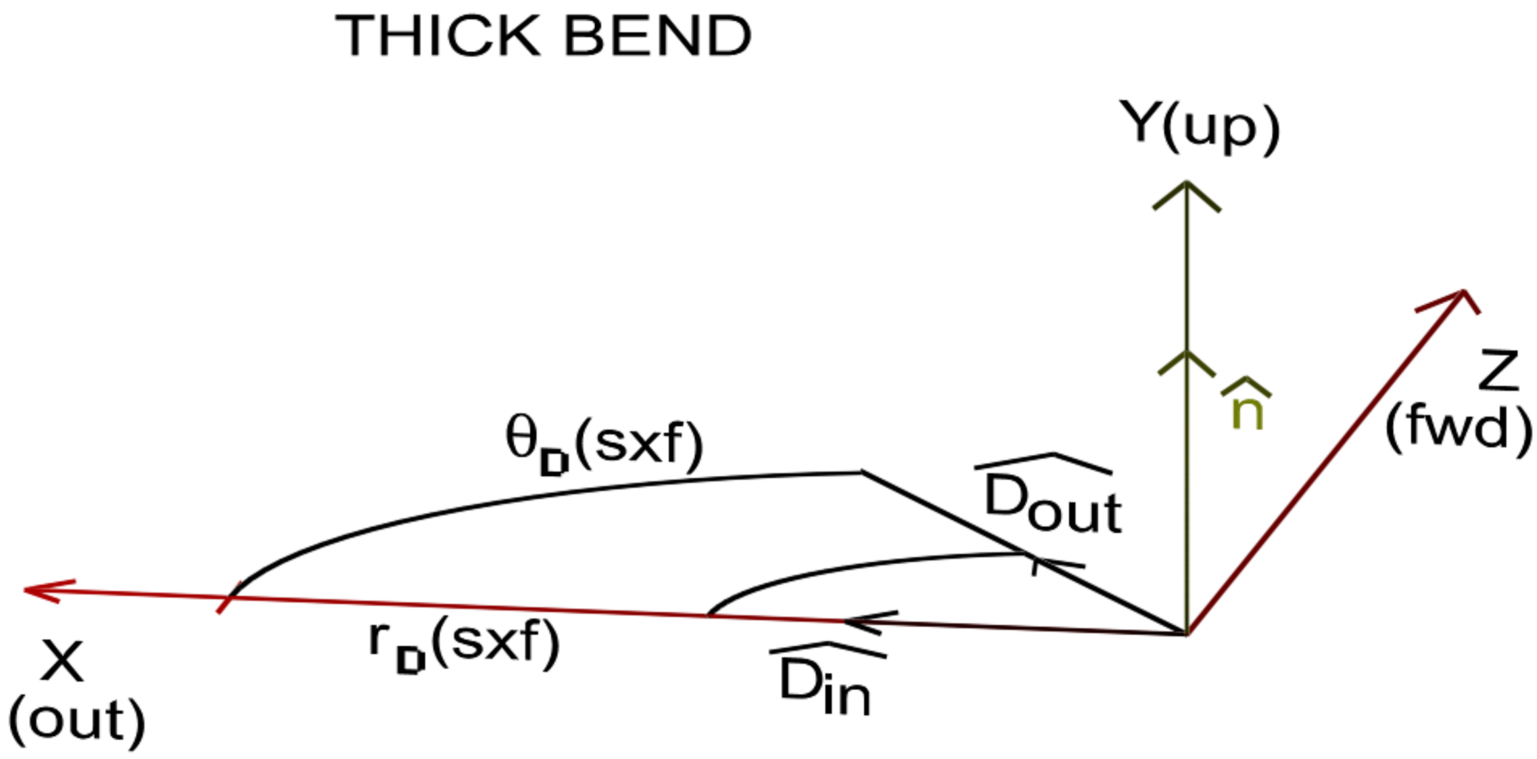

Figure: θ_D, $\widehat{\mathrm{D_{in}}}$, and $\widehat{\mathrm{D_{out}}}$ parameterize the design particle (bend plane, Y=0). They form part of the parameterization of all tracked particles, not just design. $\widehat{D_{in}} = (1, 0, 0)$. $\widehat{D_{out}} = (cos(\theta_D), 0, sin(\theta_D))$. The "$\widehat{D_{in}}$" plane (Z=0) is the same for all tracked particles. Ditto for the "$\widehat{D_{out}}$" plane. Particle deviations from the design particles values at the $\widehat{D_{out}}$ plane are what are maintained.

- Model bends as

$$\mathbf{E}(r) = -E_0 \frac{r_0^{1+m}}{r^{1+m}} \hat{\mathbf{r}} \tag{1}$$

- The relativistic inverse square law Coulomb/Kepler problem (m=1) can be solved exactly, both for orbit and spin.
- A virtual kick correction ($\sim \mathrm{m} - 1$) mimics the true m value, and maintains symplecticity.
- The r can be "ρ" when the electrodes are cylindrical.

The Hamilton-like vector is

$$\vec{h} = \gamma\vec{v} - \frac{k\gamma}{L}\hat{\theta} = \gamma\dot{r}\hat{\mathbf{r}} + \left(\gamma r\dot{\theta} - \frac{k\gamma}{L}\right)\hat{\theta} = \left(\gamma\dot{r}, \gamma r\dot{\theta} - \frac{k\gamma}{L}\right) = (h_r, h_\theta). \tag{2}$$

where k is the Coulomb factor, and L is the angular momentum. This leads to (precessing "ellipse") orbit equation

$$r = \frac{\lambda}{1 + \epsilon\cos\left(\kappa(\theta - \theta_0)\right)}. \tag{3}$$

Here (r, θ) are the polar coordinates in each particle's bend plane. The origin for θ is in the $\widehat{D_{in}}$ (Z=0) plane. λ is the "mean" particle radius. These are all particle specific parameters, known from the tracked entering conditions. Note that κ is the "betatron phase advance" in accelerator physics jargon (derived here via this Muñoz approach).

$\vec{p}(r, \theta)$ can be solved for.

This exact solution guarantees symplecticity.

18 Time of Flight

The exact time of flight equation,

$$\frac{dt}{d\theta} = \frac{r}{Lc^2}(k + r\mathcal{E}), \tag{4}$$

is integrable in closed form.

There is a numerical impediment imposed by this Muñoz absolute radius approach. Careful treatment of the formulas is necessary to avoid loss of precision.

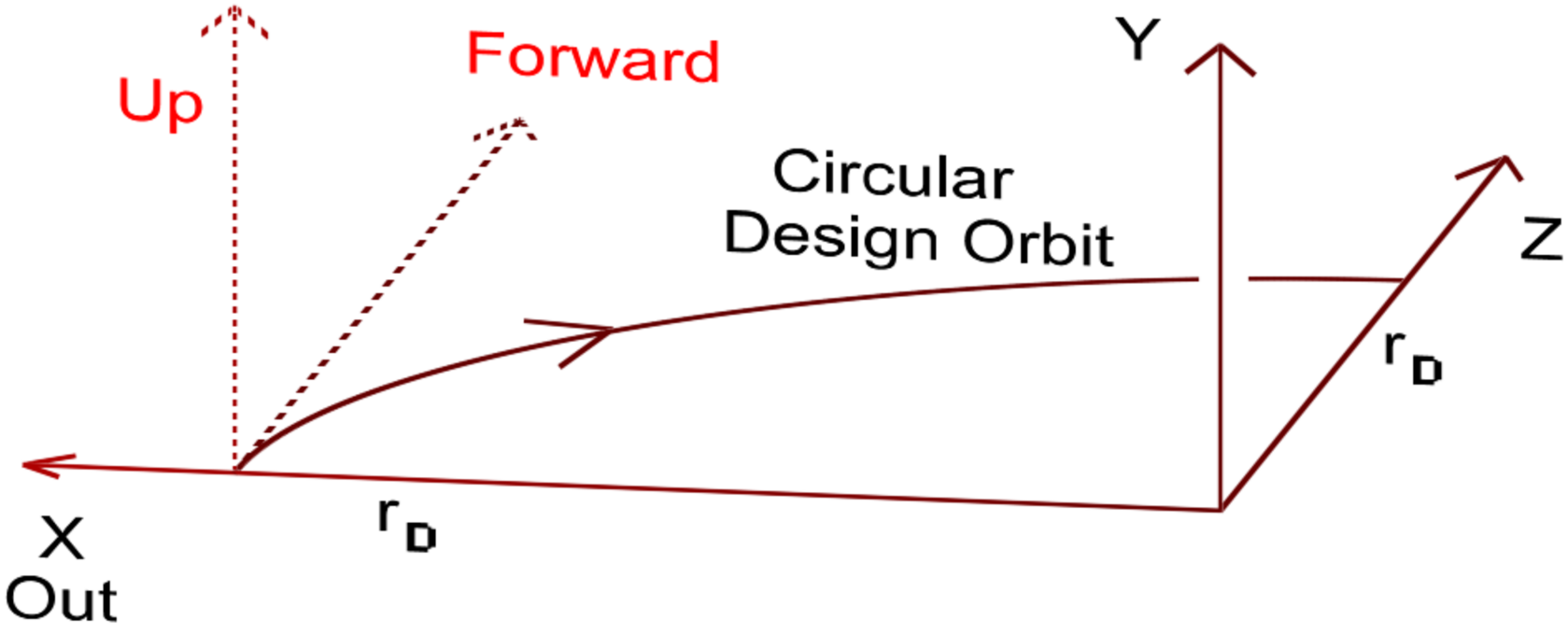

Figure: Absolute/Global/Remote coordinates versus local coordinates.

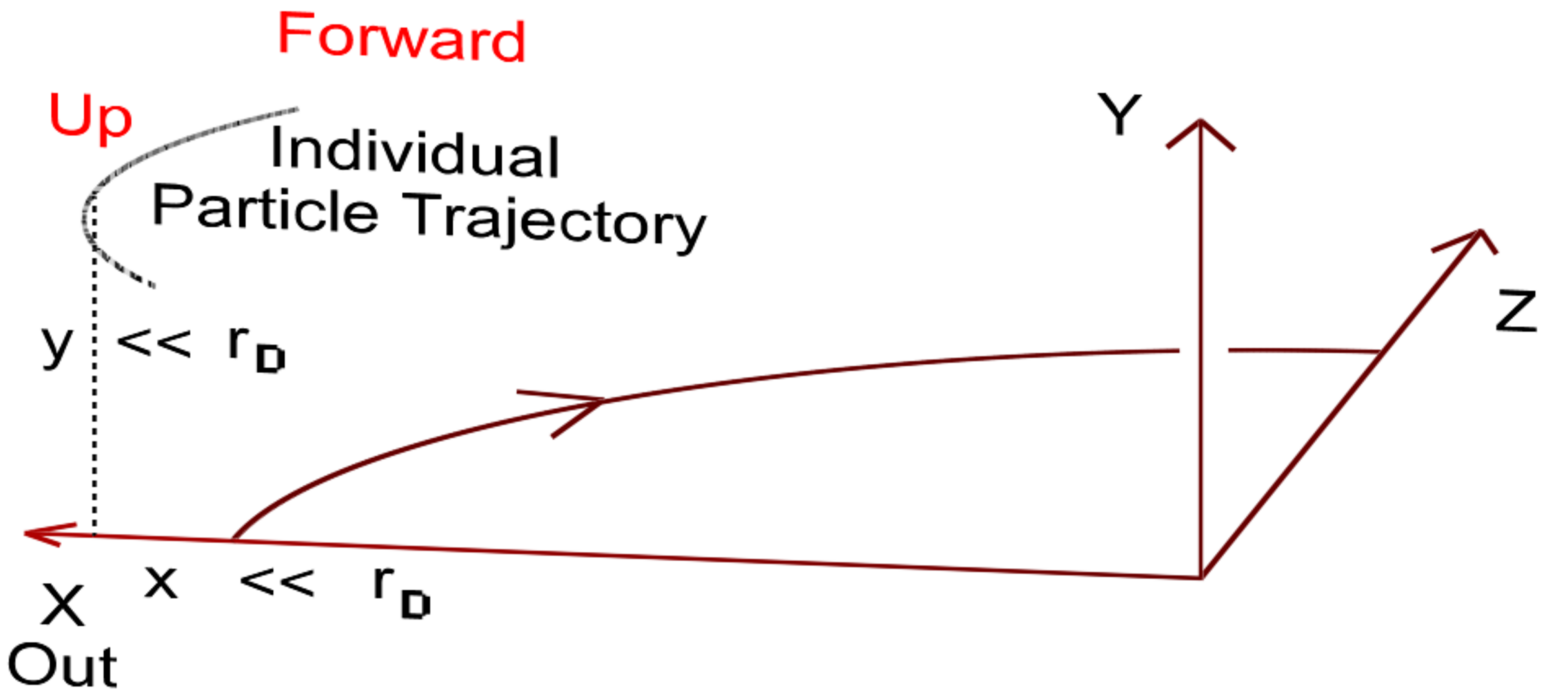

Figure: Individual particle (remote) coordinates near bend entrance. The variable x should probably be symbolized dx_0 here (0 for initial, d for deviation). The shorthand used here is typical in the accelerator world. Unusually, the quantity $r = r_D + x$ (remote origin) is meaningful in the Munoz theory and code implementation. See best practices.

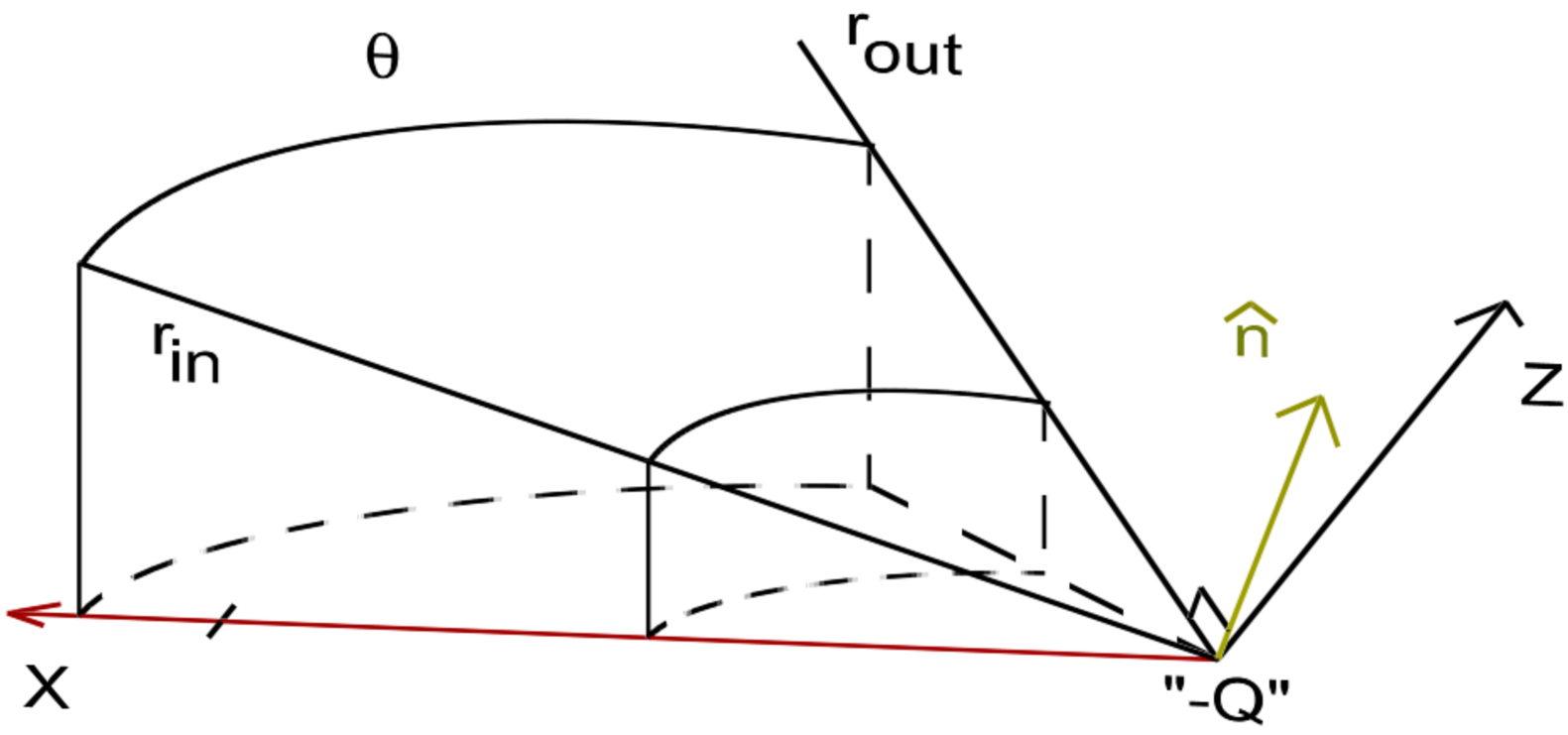

Figure: Great circle angle θ is given by $cos(\theta) = \widehat{r_{in}} \cdot \widehat{r_{out}}$. $\widehat{r_{in}}$ is known from tracking. θ_D is the projected angle (Thick Bend Figure), and is the same for all tracked particles. $\widehat{r_{out}} = a\widehat{D_{out}} + b\hat{y}$. $\widehat{D_{out}} = (cos(\theta_D), 0, sin(\theta_D))$. Then a and b can be solved for to give θ.

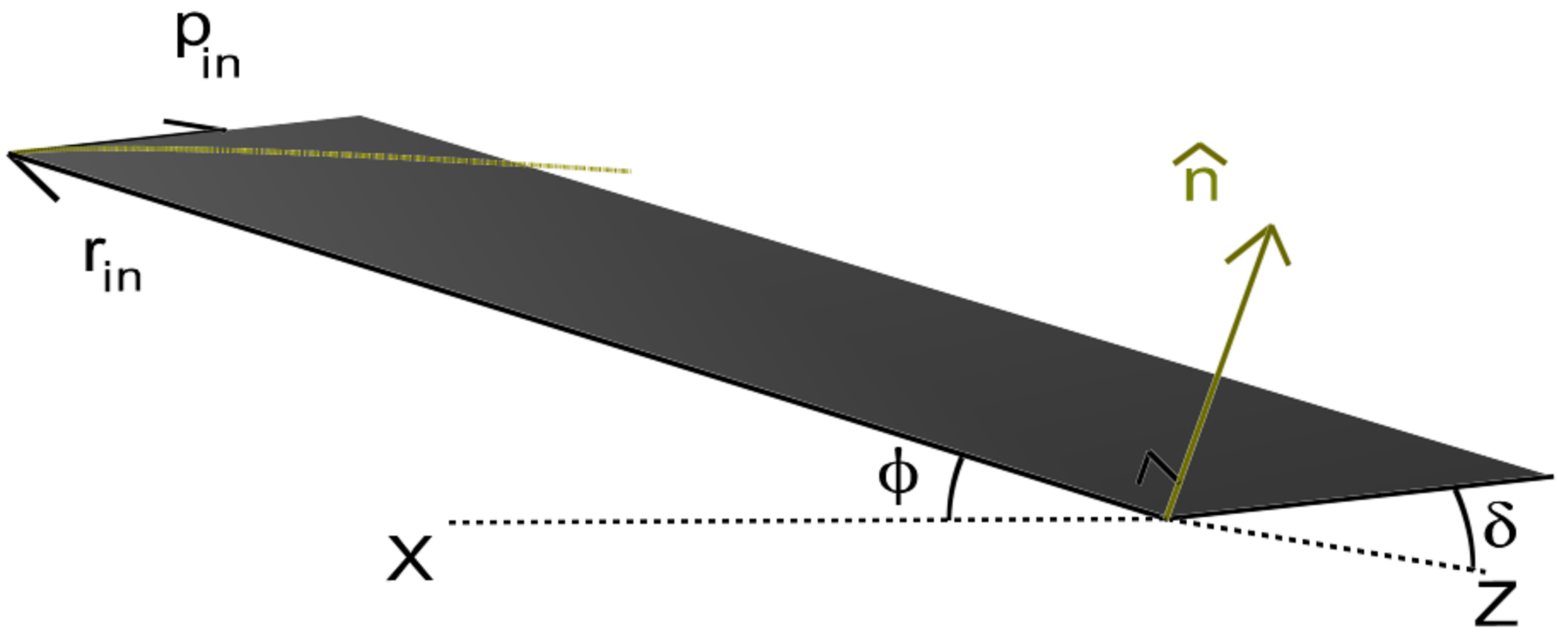

Figure: Another hopefully orienting view of the bend plane. $\widehat{r_{in}}$ points along X but can have a small Y component. $\widehat{p_{in}}$ points along Z but can have both a X componentand a Y component. This "tipped" plane is exaggerated, being much closer to design plane $\mathrm{Y} = 0$ in practice.

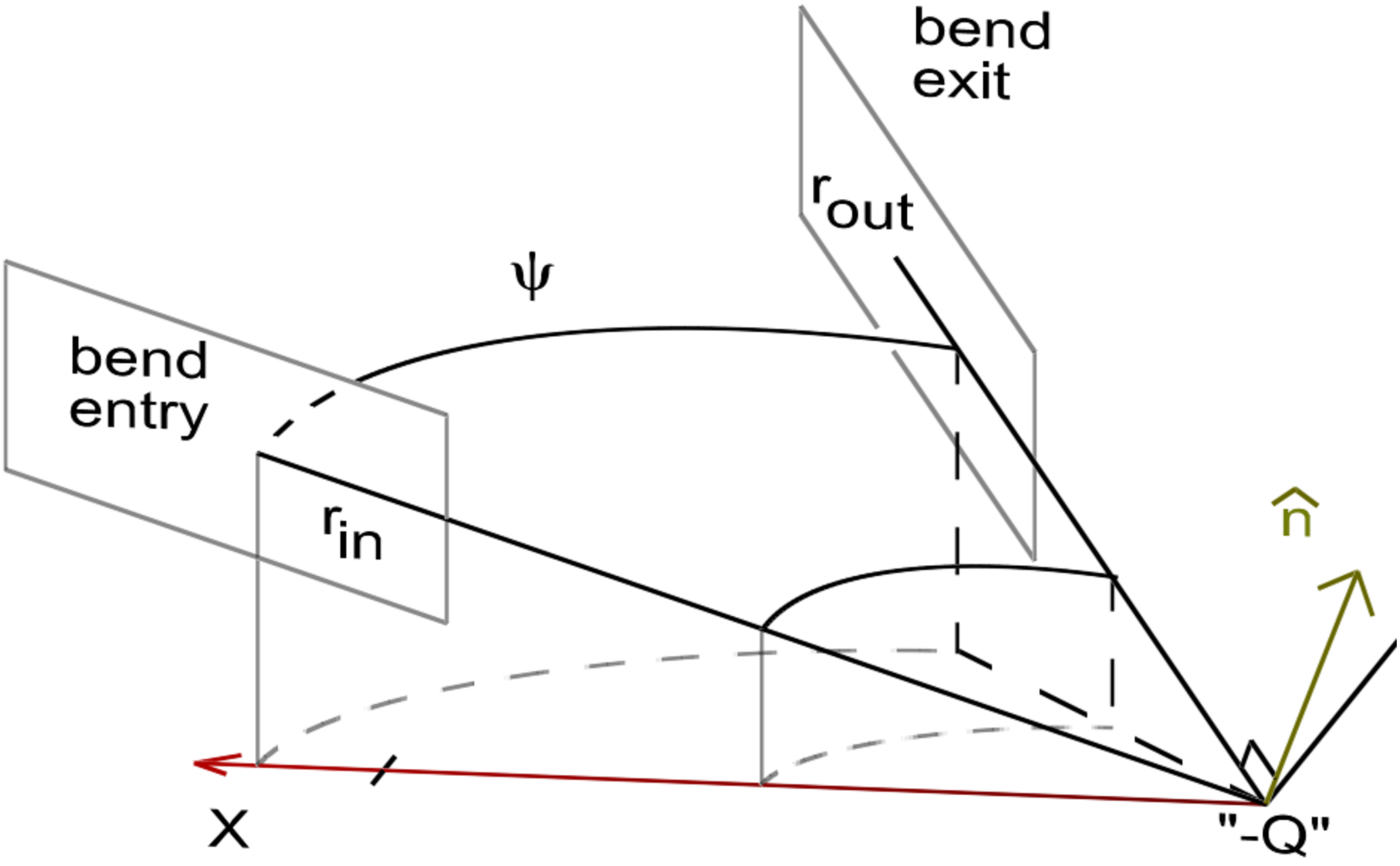

Figure: Another hopefully orienting view emphasizing the entry plane and exit plane common to all particles, not just design. The bend plane, a somewhat new/unfamiliar construct, is different to these rbend/sbend boundaries.

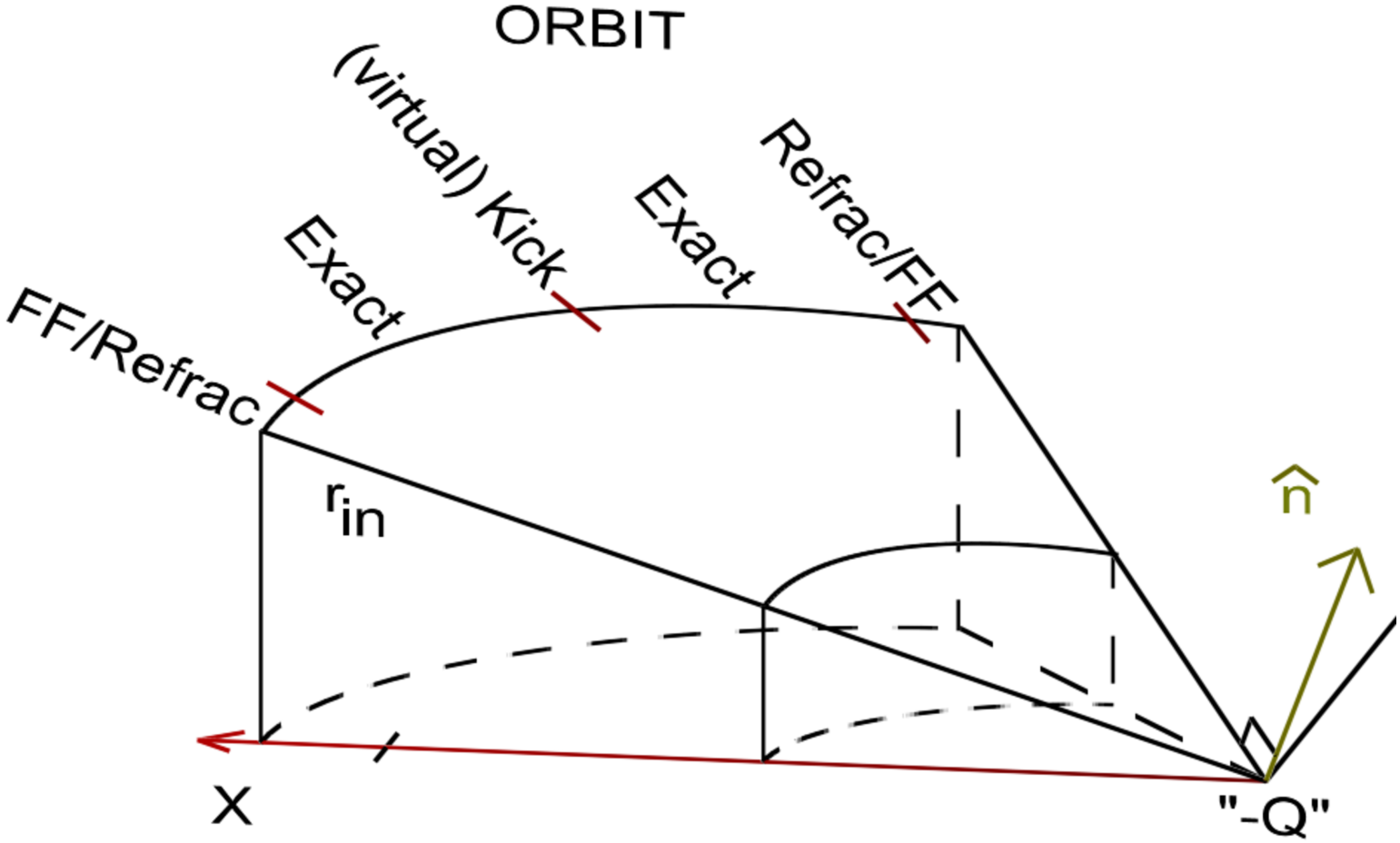

Figure: Schematic view of the sequence of software modelled physical regions in a bend for orbit propagation. UAL architecture facilitates this software sequencing for arbitrary algorithms, bunches, and lattices. A seamless matching of software interfaces to physical regions is enabled. Overall Performance is basically proportional to the number of elements in the lattice.

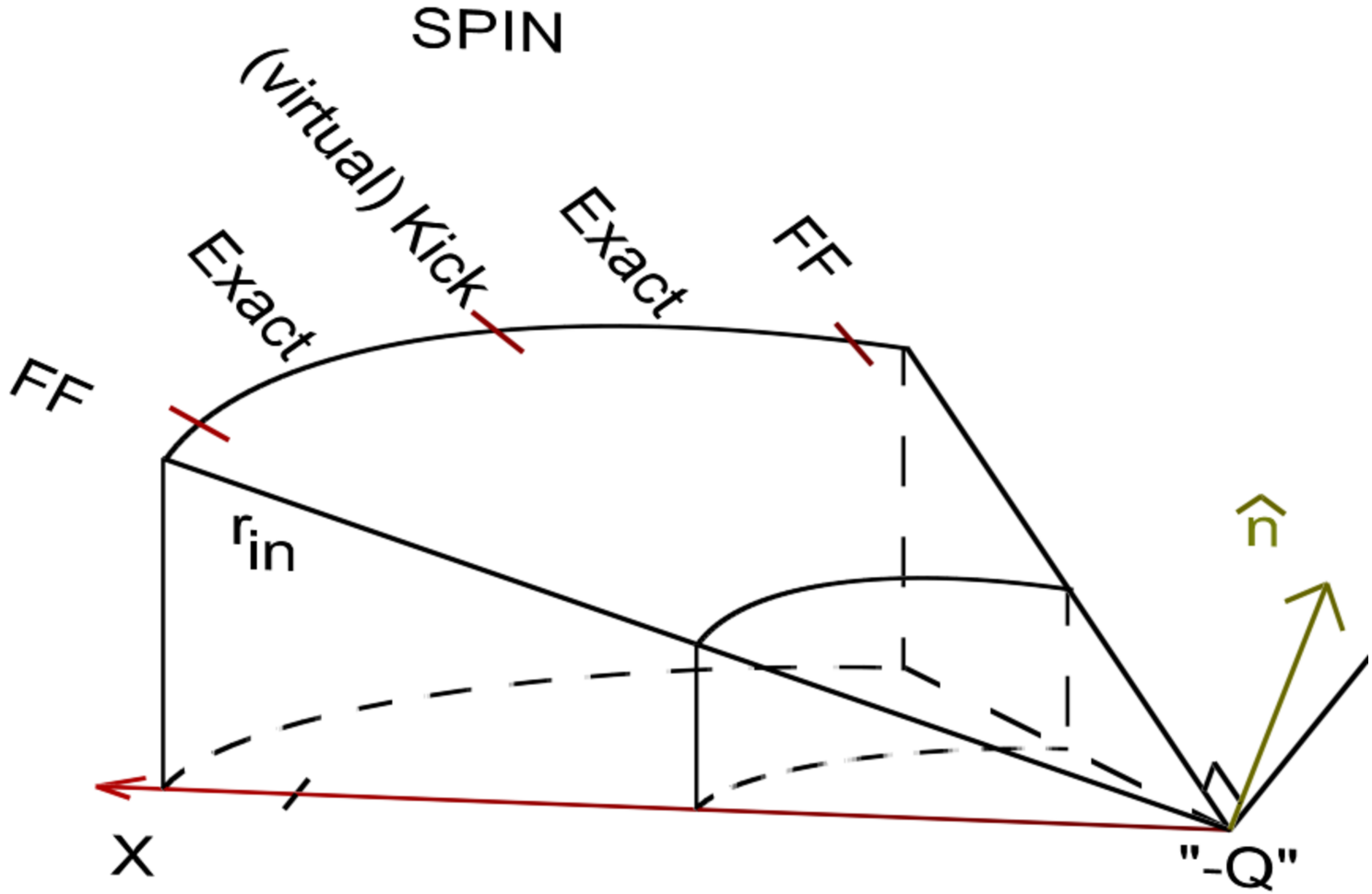

Figure: Schematic view of the sequence of software modelled physical regions in a bend for spin propagation. Spin precession in bend region is the exact solution of the BMT equation. Spin precession in the fringe field is subsumed into the bend precession.

Entry (in implicit)

- Subscript - denotes just before FF, subscript + denotes just after FF.

$$x'_-{}^2 = \left(\frac{dx}{dz}\right)_-^2 = \left(\frac{dx}{dt}\right)_-^2 \left(\frac{dt}{dz}\right)_-^2 = \gamma^2 m^2 \left(\frac{dx}{dt}\right)_-^2 \left(\frac{dt}{dz}\right)_-^2 / \gamma^2 m^2$$
$$= \frac{p_{x-}^2}{p_{z-}^2} = \frac{p_x^2}{p_{z-}^2}!$$

because p_x unchanged (no lateral/transverse force).
x, y, p_x, p_y, $\mathcal{E}$ are also unchanged. Thus $\xi = \frac{x}{r_D + x}$ is also unchanged.

- In pseudo code

$$p_-{}^2 = e^2 - m^2$$
$$p_+{}^2 = (e - E_D r_D \xi)^2 - m^2$$

allows p_{z+} to be solved for

- so that

$$p[1] = x'_+ = \frac{p_x}{\sqrt{(p_{z+}^2)}}$$

- Thus p[1] is the only PAC::Position update for the fringe field.
- Even though p_z changes, it does not update PAC::Position.
- Angular momentum, $\vec{L}$, changes, however.
  Lx = y*pz-z*py
  Ly = z*px-x*pz
  Lz = x*py-y*px
- The y component, L_y, has a nice formula. It is maintained as a global, because angular momentum is an absolute/remote construct, hence somewhat unfamiliar.
- The x, z components, L_x, L_z, are not maintained as globals.

▶

$$p[3] = y'_+ = \frac{p_y}{\sqrt{(p_{z+}^2)}}$$

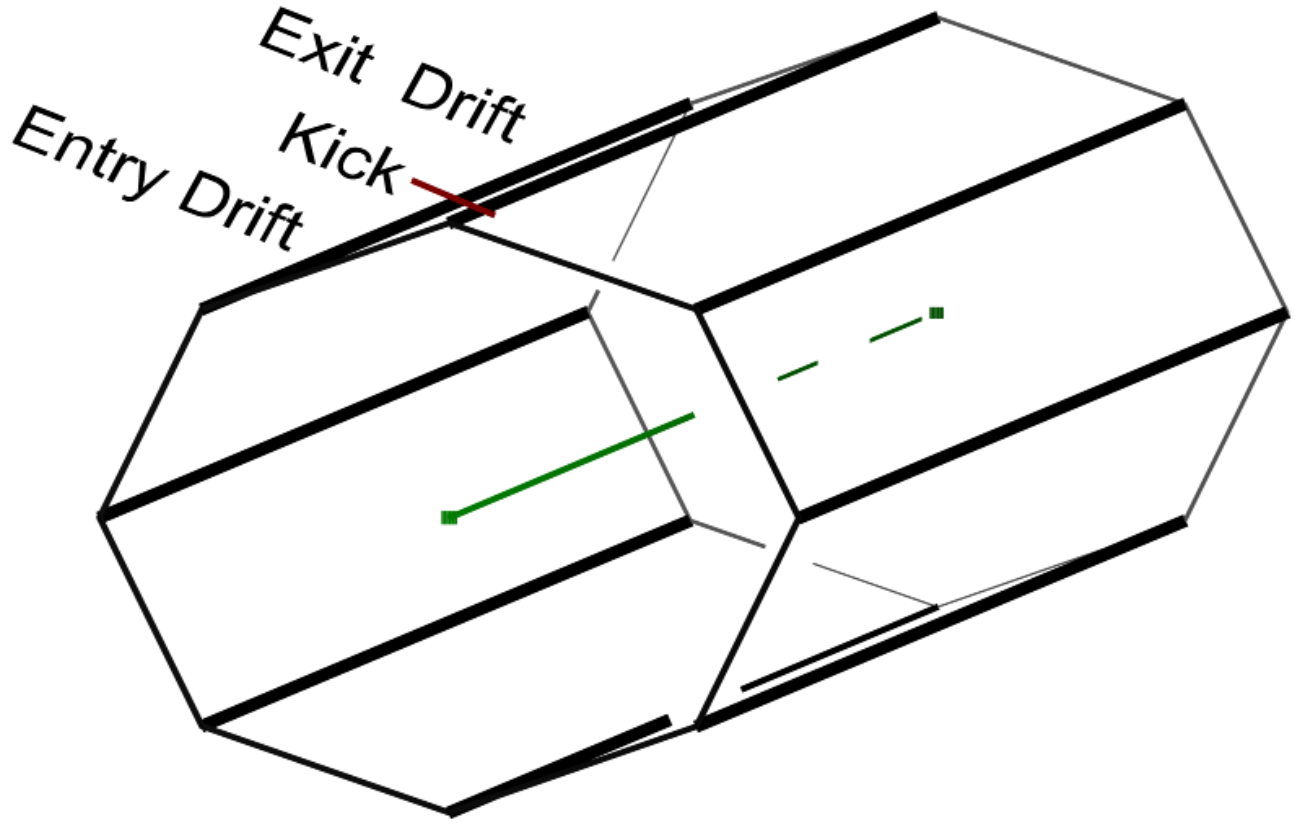

Figure: Multipole squirrel cage depiction. Schematic view of the software modelled sequence of regions in it ("drift, kick, drift"). The drifts are thick. The multipole effect is treated as a delta function ("thin") impulse. The multipoles tend to be logically specified as physically very thin in the sxf file. Design orbit (straight line through center) in green.

THIN QUADRUPOLE:

$$\begin{pmatrix} x_{out} \\ x'_{out} \\ y_{out} \\ y'_{out} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -q & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & q & 1 \end{pmatrix} \begin{pmatrix} x_{in} \\ x'_{in} \\ y_{in} \\ y'_{in} \end{pmatrix} \tag{5}$$

PAC::PacElemMultipole* mult

q = mult->data[2]

31 Thin Multipole Equation II

THIN SEXTUPOLE:

$$\begin{aligned} x_{out} &= x_{in} \\ x'_{out} &= x'_{in} - \frac{1}{2} S \left(x_{in}^2 - y_{in}^2 \right) \\ y_{out} &= y_{in} \\ y'_{out} &= y'_{in} + S \, x_{in} y_{in} \end{aligned} \tag{6}$$

PAC::PacElemMultipole* mult

S = mult->data[4]

32 Thin Multipole Equation III

THIN MULTIPOLE:

See page 116 - 119 in N. Malitsky and R. Talman
TEXT FOR UAL ACCELERATOR SIMULATION COURSE

`http://code.google.com/p/ual/source/browse/trunk/examples/ETEAPOT/legacyAndFoundation/CornellUSPAS.pdf`

The table on page 119 makes contact with the MAD-8 Documentation

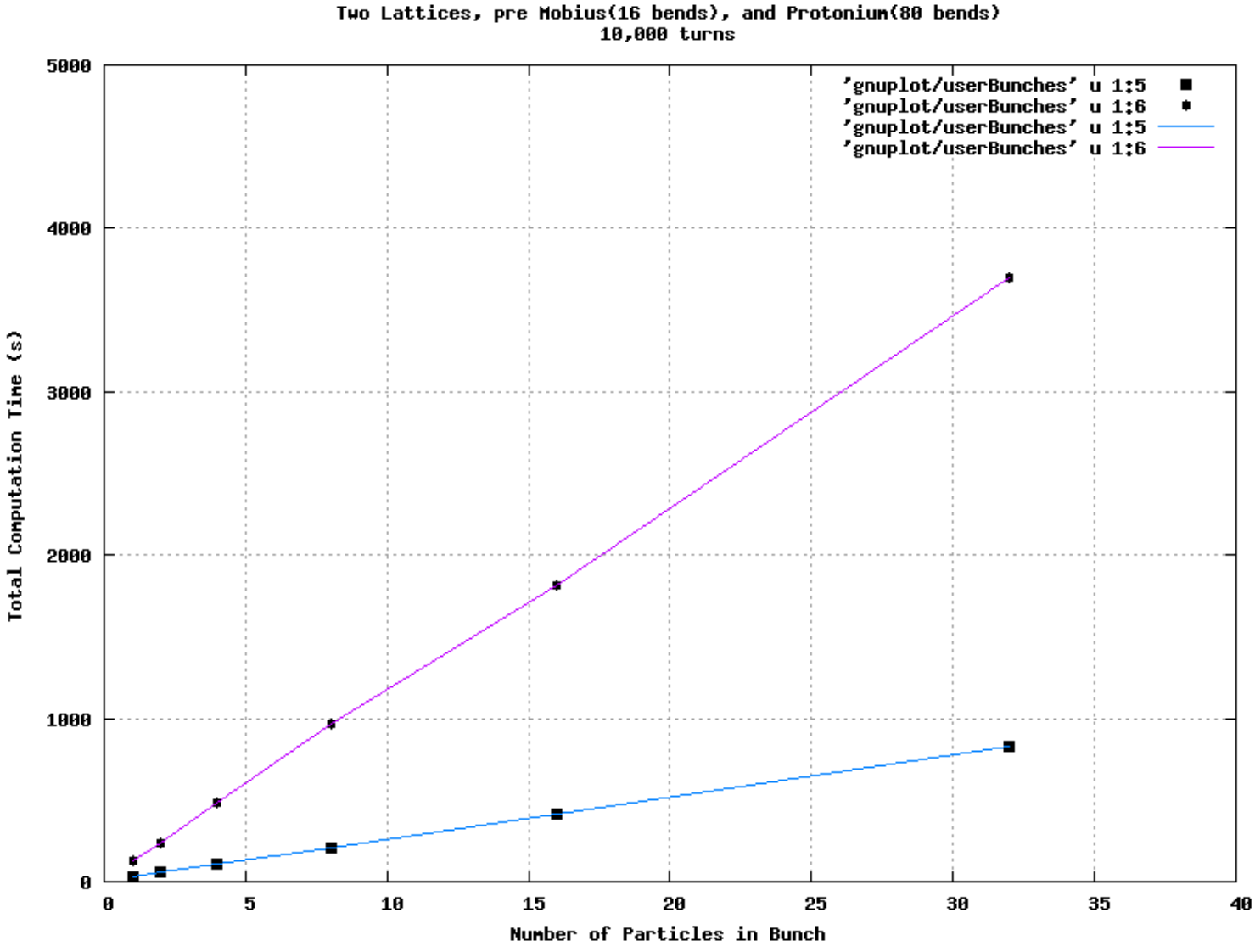

Figure: Near strict proportionality between run time and number of particles.

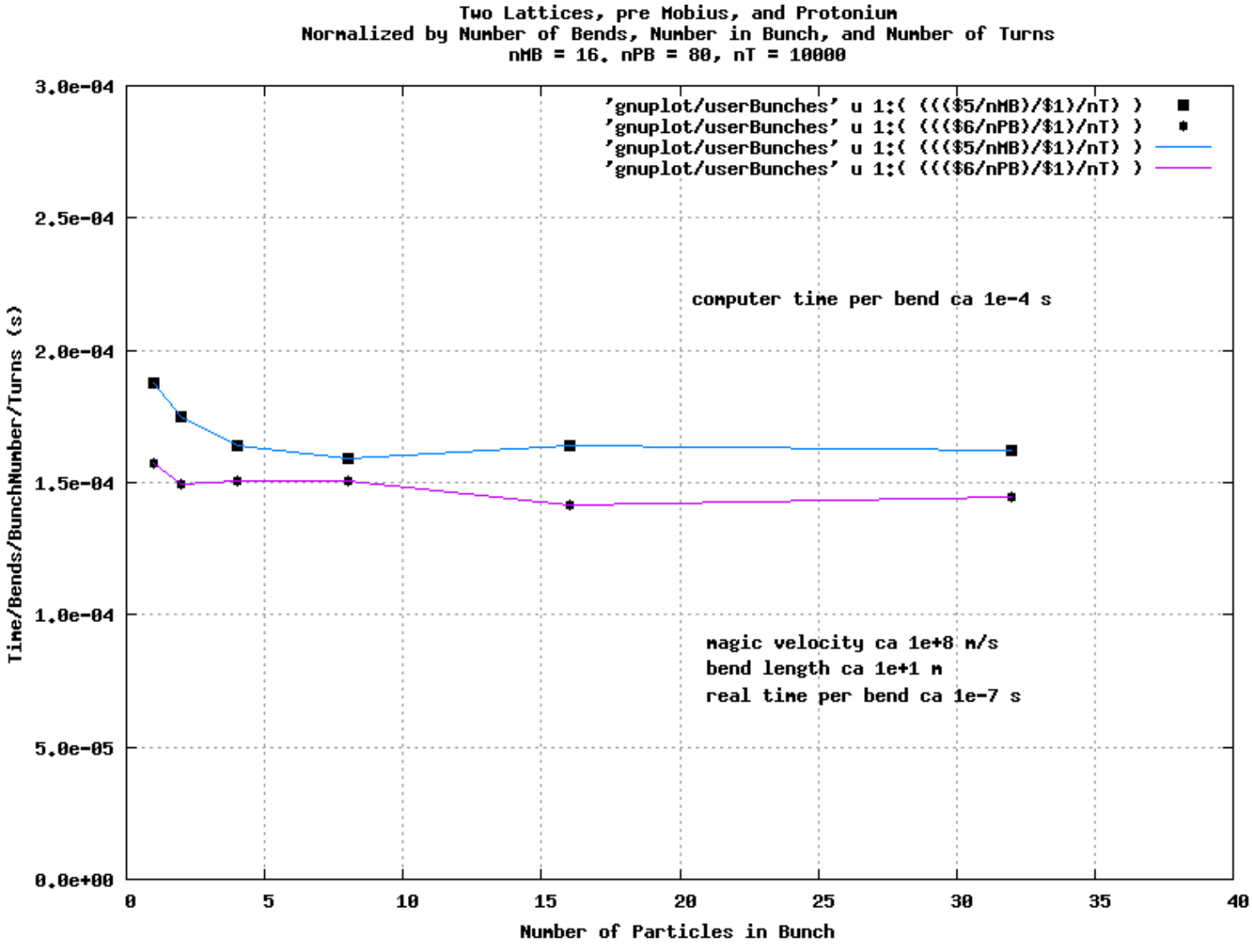

Figure: Order of magnitude parallel processing estimation.

1. ETEAPOT is a "streaming" application with few branches, low memory requirements (everything might be cacheable on the "local core"), low i/o, and few interparticle interactions.
2. Circa 10^3 factor between real time and computer time.
3. Can pick up a factor of ca 4 on a simple COTS laptop.
4. Rack of 10 units for ca \$10K would give a factor of ca 40!

Multicore cluster/farm with one thread (user bunch particle) per core. $N_{bend} = 16$, $N_{turns} = 10000$, $N_{time} = 10^4$, $N_{ptcls} = 32$, $T_{comp} = 820$ s.

$$\frac{\text{computational time}}{\text{bend} \cdot \text{particle}} = \frac{820}{16 \cdot 10^4 \cdot 32} = 0.16\,x\,10^{-3}\,s. \qquad (7)$$

Computer clock speed was 2.5 GHz.

For the ring used, circumference is 277 m at the magic velocity 1.8 x $10^8 \frac{m}{s}$, the ratio of computational time to real time

$\frac{t_{comp}}{t_{real}} = \frac{820/(32 \cdot 10000)}{277/1.8x10^8} = 1.7x10^3 \approx 10^3$.

37 Parallel Processing: GPU

1. ETEAPOT might be a good candidate for GPU (many "slimmed down" cores) exploration.
2. ETEAPOT: PAC::Position(6) x ca 10 - 100 standardized particles per user bunch.
3. We favor the CPU route with a cluster of conventional linux processors with routine networking.

1. An up front awareness of the steep learning curve associated with UAL development and usage.
2. Coding matches the governing physics document precisely. Accompanying diagrams are essential.
3. Reduce clutter, generally. This includes both code and directory structure.
4. Capacity for counter circulating beams should be anticipated from the start. Ditto for both signs of the particle charge.
5. Developer and the user on the same page (e.g. diagram) from the start. A common symbology, for example signed angles, can then be established.
6. Relevant parties maintain some sense of the differences between their personal development files and the version controlled files. This is where the branch construct/motivation arises in version control, and is to be avoided.

39 Best Practices II

1. User manifest directory pooling all the configuration files.
2. Reducing clutter in examples directories, particularly. A simple/single working recipe is a corollary. This then gives three files, the main C++ source file, Makefile, and recipe, in the top directory.
3. Units: MKS, traditional accelerator, "Munoz".
4. Related to this is local and remote, coordinate systems and equations, awareness.
5. Consistent Symbolization. Consistency in notation.

Best Practices III

1. Name collision and disambiguation awareness. THE UAL architecture, with it's namespaces, facilitates this.
2. Avoiding direct usage of r_D in the equations can save 3 or 4 digits of accuracy (combining two quantities e.g. r_D and x differing in magnitude by a factor of 1000+ is a bad idea).
3. Reference the exact UAL version number used for the results in any documentation and/or reports
4. Theory and diagram(s) that overview the programmatic lattice sequencing for each element type.
5. Comments in the code identifying the section in the governing physics document they are following.

41 Summary

1. Exact analytic solution in idealized lattice is fast, and guarantees symplecticity.
2. Symbolic/Syntactic computer software engineering approach facilitates moving to code release.
3. (Non interacting) multiparticle bunch structure and overall UAL/ETEAPOT architecture facilitates parallel processing.
4. The absolute coordinate approach, unconventional for accelerators, requires careful numerical treatment.

42 Potential Simulation Projects

1. Equilibrium (6+3)D beam phase space/spin distribution.
2. Determine injection depolarization.
3. Calculate full run depolarization (as a function of amplitude).

[1] N. Malitsky, R. Talman, J. Talman
Appendix UALcode: Development of the UAL/ETEAPOT Code for the Proton EDM Experiment

[2] Munoz, G., Pavic, I.
A Hamilton-like vector for the special-relativistic Coulomb problem
European Journal of Physics
27(2006) 1007-1018

[3] H. Weidemann
Particle Accelerator Physics I: Basic Principles and Linear Beam Dynamics
Springer, 1999

[4] E. Forest, F. Schmidt, and E. McIntosh
Introduction to the Polymorphic Tracking Code
CERN-SL-2002-044 (AP)
`http://cern.ch/madx/doc/ptc_intro.pdf`

[5] H. Grote and F. C. Iselin
mad_user.pdf
CERN/SL/90-13 (AP)
http://mad.web.cern.ch/mad/mad8/doc/mad8_user.pdf

[6] W. Herr
madx_tutorial.pdf
CAS 2011
http://cern.ch/madx/doc/madx_tutorial.pdf

[7] W. Herr and F. Schmidt
madx_primer.pdf
CERN-AB-2004-027-ABP
http://cern.ch/madx/doc/madx_primer.pdf

[8] madx_manual.pdf
http://cern.ch/madx/doc/madx_manual.pdf

[9] L. Schachinger and R. Talman
TEAPOT: A THIN-ELEMENT ACCELERATOR PROGRAM FOR OPTICS AND TRACKING
Particle Accelerators, 1987, Vol. 22, pp. 35-56

[10] N. Malitsky and R. Talman
TEXT FOR UAL ACCELERATOR SIMULATION COURSE
http://code.google.com/p/ual/source/browse/trunk/examples/ETEAPOT/legacyAndFoundation/CornellUSPAS.pdf
Select "View raw file"

[11] N. Malitsky and R. Talman
UAL User Guide
December 20, 2002
Appendix C
http://code.google.com/p/ual/source/browse/trunk/examples/ETEAPOT/legacyAndFoundation/24937.pdf
Select "View raw file"

[12] N. Malitsky and R. Talman
User Guide for UAL (C++ Interface)
September 2008
Appendix C
http://code.google.com/p/ual/source/browse/trunk/examples/ETEAPOT/legacyAndFoundation
Select "Show details"
Select "View raw file"

[13] J. Talman
Independent Variable Forms for Coulomb Relativistic Radius: Targeting Time of Flight Results.
March 30, 2014
http://code.google.com/p/ual/source/browse/trunk/examples/ETEAPOT/munozUtilities/relativisticRadiusVsTheta.pdf
Select "Show details"
Select "View raw file"

[14] Y. Semertzidis

[15] V. Lebedev

[16] http://en.wikipedia.org/wiki/Software_release_cycle

[17] http://www2.hawaii.edu/~ztomasze/teaching/ics211/2012fa/lecture/02A.html

[18] http://en.wikipedia.org/wiki/Frenet-Serret_formulas