ETEAPOT Review, Outlook, And Travelogue

J. Talman

August 29, 2014

Abstract

ETEAPOT (Electric Thin Element Accelerator Program for Optics and Tracking) has been an effort to clone/mimic TEAPOT/UAL (Unified Accelerator Libraries) functionality, but with electric focussing elements, rather than magnetic. The development has been uneven, with some notable bugs. It has shown potential, however. Presumably, these bugs have been fixed, or at least improved. Recently spin propagation, the main original impetus, has been added. Ambiguity in the orientation/sense (clockwise, CW, and/or counter clockwise, CCW) of simulated propagation has arisen. Lessons have beeen learned. This document discusses these issues, emphasizing the background and terminology. This allows a more precise quantitative demonstration of the current codes accuracy, precison, and potential. Current code references are made. Possible best practices for a possible next code cycle are identified and, when possible, articulated. This is one of the main goals, here. This might make the format of this report a little less linear/sequential than the norm. Also, code considerations sometimes get mixed with physical considerations. A possible first best practice is that capacity for counter circulating beams should be anticipated from the start. A second is getting the 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. This is easier said than done, however. For example, giving each lattice element its own "bend plane" is an orientation that only developed after a year or so of R & D. A second example is that TEAPOT has all "thin elements", hence a general "multipole plane", while ETEAPOT makes bends "thick". This requires bends to be treated quite differently than other multipoles, ideally. Quite a few (20) hopefully orienting diagrams (travelogue) for these issues and the overall subject are included.

Contents

1 Introduction

An effective platform for modelling electric focussing element accelerators is clearly of interest. Particularly for EDM (Electric Dipole Moment) experiments.

ETEAPOT is one such platform. It is still in development. However it has been through several years of critique/feedback probably constituting something near one full development cycle.

This document is intended to substantially complete such a development cycle. It may serve to facilitate starting the next cycle.

Unusually, this first pass has involved mostly only one developer (me). Furthermore, this involved several staggered stints with substantial gaps inbetween. Sometimes I was working full time, and sometimes I was working part time. Also the main user (Richard Talman) was remote during most of this. A best practice recommendation is to have sustained full time interaction between the developer and the user in a first development cycle. This can be relaxed substantially in subsequent cycles. The most fruitful development environment is likely to be one where "the developer is the user". Tight interaction between the developer and the user is almost equivalent ("next best thing"), and might even be more robust in that cross checking can be done more reliably and much more quickly. It might be a double edged sword, however, because this may cause the individuals to give less priority to their own work (which they're probably more confident of). Also, it is easier to "cut corners" on documentation, relying on verbal communication. There was a powerful "architecture" that handled bunch size and sequencing available (UAL). This was grafted onto by me, but any further architecture tended to be ignored. The formulas tended to be developed on the fly, so an "open ended" develepment style that accomodated this was what I tended to favor. I've come to view this first develoment cycle work as a prototype. The code is functional, and "near working", but is not very accessible or user friendly.

A best practices recommendation is to have all formulae, with accompanying diagrams, available up front. This is easier said than done. It may be that it is only practical in a second (and subsequent) development cycle. At the very least, maintaining an overall theory document is an all but essential best practice. In "theory", the theory document and the code are in exact agreement. In practice, this is often not completely achieved! It should certainly be strived for. The "get it working" mantra usually seems to govern!

Another all but essential best practice is that all 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.

Of course the scope of the project is critical for these heuristics. This was an essentially two person effort. "Practical" practices will change for much larger teams. Experience level, and total experience will play a role. Project criticality, modest here, can dictate the whole effort schedule. It may be that it is unreasonable to plan for significant new software capability in compressed time frames!

This document overviews the ETEAPOT platform starting with some terminology rambling, and accelerator coordinate systems generally, and ETEAPOT coordinates specifically. Sense of propagation (CW or CCW) is inextricably linked. For counter circulating beams, one needs both. Also, both signs of the particle charge often need to be dealt with.

It then moves on to equation analysis, eventually favoring "local equations" which are the norm in magnetic propagation simulation work (e.g. TEAPOT/UAL), and for accelerator physicists, generally. "Remote equations" have arisen, possibly unwisedly, in this first development cycle.

A brief code organization section summarizes cycle one, and looks at cycle two. This is very complex code. The user has a lot of responsibilities (parameters). A "User Manifest" of 6 or so files is identified. "Recipes" precisely specifying example simulations are all but essential to get started. Even an experienced user is likely to need something similar. One of the main goals of a next development cycle would be to improve this error prone process. Reducing clutter in each of the main examples directories is another. These probably qualify as best practices. ETEAPOT and SXF_Tracker deviated from this TEAPOT/UAL practice partly because the developer (me) wasn't aware of it, and partly because the current codes architecture didn't have the right libraries. I probably should have exploited the very powerful accelerator propagator description format (apdf) more. There is a steep learning curve associated with UAL development and usage. An up front awareness of this is another possible best practice.

Spin issues are then focussed on as a kind of culmination. Spin simulation and tracking to model fundamental particle (e.g. deuteron, electron, muon, proton) properties is regarded as all but essential to confidently build an actual accelerator for their measurement.

A brief Conclusions section wraps it up. Most of the results have already been touched on at this point.

2 Terminology Preamble

Clearly a coordinate system must be chosen to model physics. Their units must be identified. A possible best practice is the observation that units, with their implied magnitudes, turn out to be a significant hassle. Of course there are MKS units. Then there are "traditional" accelerator units, which are near MKS, but enough different to require careful cross checking. In this first development cycle, "Munoz" ([1]) units played a large role. They are "new/unfamiliar", and may, or may not, play a role in any next development stage. Probably obviously (implicit best practice), a consistent symbolization for all this should be defined from the start, and then adhered to. An overview of the units issue is in section 4.1 of [2]. The coupling between coordinate system, units, and modelling may have been unusually strong in this first development cycle.

Perhaps storage rings are "global" constructs with a "captured beam" giving readily distinguishable/identifiable orbits as the central concept. Perhaps accelerators are "local" constructs with only an element by element reality. Probably they are both. A complex system not amenable to simple characterization! Accelerators have a linear/sequential nature, however. Some of the primary theory for this work is [1]. This can be probably be characterized as global, or at least remote, with its near unique origin applying to several bend elements at once. Real elements will have imperfections partially negating this "degeneracy". Furthermore, the code implementation uses only the local information of the tracked particle coordinates entering an element, and the elements physical properties. It seems as though lattice designers (accelerator physicists) tend to have a "package" view of several sequential lattice elements that has net focussing (e.g. FODO) in all necessary coordinates). This structure then must be replicated to "flesh out" the full circumference. Achieving a "stable lattice" is "the goal" and is quite difficult, generally.

I have now used three adjectives: global, local, and remote. I try to clarify these terms in the following.

Some people point to the power supply as always causing some overall coupling in coordinates. Of course, part of a stored beam is a large power drain! This seems global, in some sense, to me as it involves the general principle of increasing overall (system/global/universal) entropy.

Another units issue is that spatial variables and momentum variables have different units and so displaying them in a single figure is somewhat arbitrary. I do just that, however, in my "travelogue".

3 Accelerator Coordinate Systems Background

References [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12] all discuss accelerator coordinate systems. They are usually right handed, as are mine. CW and CCW propagation can both be present.

General coordinate systems in physics have a large literature. Perhaps, starting with special relativity, more attention was paid to precise specifications.

Accelerator coordinate systems are simpler in that all observers have the same clock (they are at rest with respect to each other). This is probably the only place where the adjective simple can be used, however.

The "event" associated with beam injection is convenient in that it can serve to define the "global coordinate system". Thus the origin of this global system is conventionally taken to be at the injection "point". An inert lattice element at this origin is another (code) standardization. Of course there are additional details (to follow)!

Beam injection is important, also, in that it is relative to a magnetic system, typically. Thus "translating" between magnetic units and electric units becomes a source of ambiguity.

Having a neutral element("drift"/"marker") at the origin means the beam doesn't have to be injected into a strong field. This is probably conducive to capturing more beam particles with more favorable spreads, so to speak.

One has to be careful to distinguish code conventions (e.g. marker at origin), from physical/practical conventions (e.g. inject into zero potential regions). This latter, potential regions, is associated with electric bends, not magnetic.

The terminology "outside" active elements then arises. There is no potential energy of any kind, so the energy is just mechanical. This is opposed to "inside", where any particle has had to climb/fall some sort of potential well. This is especially apt for the main bending elements in this electrical case.

The linear direction of accelerator beam propagation defines a natural coordinate ("tangent", or "forward"), and some sort of "global forward" might be used to specify e.g. beam polarization (or vice versa). Beam polarization is a collective phenomena, however.

Diagramming an accelerator coordinate system is difficult. [3] has such a diagram, Figure 1 here, of what I'll call a "local coordinate system" where all particle coordinates are "small" (relative to bend radius for the spatial transverse components x and y, relative to design momentum p_D for the momentum transverse components p_x and p_y). Note that it depicts CW propagation, and is right handed. It uses ρ for bend radius, while I tend to use r_D. Mild objections are that $\rho >> x, y$, and the "3D" nature of the process is possibly somewhat obscured. Figure 2 is from [5], and is another local coordinate system. These two figures are all but equivalent to my mind, though the reader may want to mull them over. This is encouraged and probably I should say the reader should look them over! They are used to set the stage for the rest of this work. Figure 1 does not explicitly label the "centre of curvature", while Figure 2 does. Figure 1 labels its axes with bolded, larger, symbols,

$\mathbf{x}, \mathbf{y}, \mathbf{s}$. It then uses lower case x, y for the individual particle coordinate deviations/displacements/offsets/("signed excess") from design. It does not explicitly label s for the $\mathbf{s}$ displacement, though is implicit. This "transverse" emphasis turns out to be ubiquitous.

The X transverse motion is sometimes called "Out", and sometimes horizontal. The Y transverse motion is sometimes called "Up", and sometimes vertical.

Possibly confusing in Figure 2 is symbol s denoting arc length, presumably, and not a coordinate direction. I eventually favor Z for the "Forward" axis, but will take a little time getting there. "Longitudinal" tends to be synonymous. Again, the normalization of tangent to increasing arc length defining a forward direction is also convenient. Maintaining a design/ideal/reference/zero particle, that all other particles are referred to, can be confusing. $\Delta s = r_D \Delta\theta$ for this particle. There is perhaps a tendency to confuse the θ of [1] with the independent variable s expressed in terms of the design particles θ. The theory of [1] is two dimensional ("tipped" planar). This makes polar coordinates natural. Simulated, tracked, particles (probes) have three spatial coordinates, however. Of course they have additional momentum coordinates (three, for a total of six), as well. Name collision and disambiguation is a well known issue in computer science generally. Being careful about it, and avoiding it is a more general best practice than the application here, but I'll mention it anyway.

Possibly also confusing in Figure 2 is the designation "actual orbit". There are many actual/possible orbits in accelerator beams, simulated and/or real. A live accelerator might have 10^{10} captured particles in its beam. They will each have their own "actual orbit", different to all the others, generally. The design orbit (reference orbit), for example, is possible (though not likely exactly achievable with "shotgun" injection). Thus the designation of Figure 1, "individual particle trajectory", is maybe more apropos.

The bilateral nature of Figures 1, 2 about the "Out" (X) axis is probably useful for spatial orientation for the reader/viewer. It is probably misleading for specifying a real coordinate system suitable for integrating/simulating real beam particles as this system is unaware of the particles prior trajectory (history) other than its values at the entrance. Mostly only the offsets to design (x, y), and their "slopes" (x', y'), are relevant. Longitudinal offset s (or z) is taken to be zero, though this must be kept track of via deviation in time of arrival to the entrance. Again, this can make the distinction between s and z confusing.

Slope variables x', y' and momentum variables p_x, p_y are very simply related. $x' \equiv \frac{dx}{dz} = \frac{dx}{dt}\frac{dt}{dz} = \frac{dx}{dt}/\frac{dz}{dt} = v_x/v_z = \gamma m v_x/\gamma m v_z = p_x/p_z \approx \frac{p_x}{p_0}$. p_0 is the design beam momentum, which is occasionally symbolized p_D, where "D" stands for "Design"; the purpose is to avoid misinterpretation of p_0 as an initial value. p_z never deviates much from p_0 in a captured beam. By convention, in MAD and UAL, $p[1] = p_x/p_D = p_x/p_0$. The author believes, as a best practice for the next code generation, that p_D is preferrable to p_0, both in the code, and in the documentation. Consistency in notation, however, is perhaps the most important criteria. If one is going to label initial values with a 0 subscript, one should do so for all of them. Of course this is easier said than done, and may even overly impede progress. The code is just a tool for investigating the actual physics!

Possibly also misleading in Figures 1, 2 is the implication that particle trajectories are always near circles. This is true for the main bending dipoles (both magnetic and electric) which are the main focus here, but is not true for other important elements such as quadrupoles and drifts. Quads are usually handled as "kicks" (continuous spatial variables, discontinuous momentum variables). Drifts give straight lines. Each lattice element becomes an initial value problem for each simulated beam particle, but the trajectory is not necessarily a near circle segment. In some sense, each lattice element defines its own observer. These observers can take almost arbitrary orientations with respect to each other, though they tend to be simple rotations about Y, and there's the closed design orbit requirement. Design orbits are taken as in the $Y = 0$ plane. With 100 lattice elements not atypical, and a 1000 not implausible, this is a lot of observers!

The exact lattice specification for a specific accelerator is contained in an sxf (standard exchange format) file. The apdf mechanism and the sxf mechanism give UAL users a lot of flexibility, generality, and power.

An observer (laboratory/gather) is also necessary to combine all these results. This could be called a "global" observer. Tracking programs typically are geared for such a global coordinate system. "The programmer is the global observer".

By always maintaining the total deviation from design, such an implicit global observer can be effectively defined, especially on a turn by turn basis. The design orbit is usually known at each element entrance, so that the local observer coordinates don't have to be spelled out explicitly. These individual axes don't have to be explicitly named (labelled) in diagrams. Thus "deviation coordinates" x, y can be envisioned for each, and all, lattice elements. Possibly, conceptually, their label should be subscripted so as to reflect their initial value nature e.g. x_0, y_0. Possibly, conceptually, their label should be modified so as to reflect their differential value nature e.g. dx, dy. Neither of these will be done here. One reason, not the most important one, is that while y is always small in a captured beam, x can sometimes be comparable to the design radius r_D, $x = r_D + dx$. Possibly the $d\vec{r}$ in Figure 2 should be $d\vec{r}\,(x_0, y_0, ...)$, or $d\vec{r}\,(dx_0, dy_0, ...)$! Really it should be $\Delta\vec{r}\,(...)$ to reflect the finite nature of any tracking simulation. This is articulated in more detail in the following sections.

There is generally a fringe field associated with each elements entrance. Then labelling parts of a particles trajectory as "just" preentrance ("-") and just post fringe field, just inside the entrance ("+") becomes a possibility. One can use subscript "in" rather than "0" to identify initial values. Thus one might label the initial probe position as $\vec{r}_{in}$ or $\vec{r}_0$. This is just post

entrance fringe field here. Physically, the exit fringe field is likely to be all but identical to the entrance fringe field. Thus these comments apply there as well.

The distinction between "thin elements" and "thick elements" arises. Thin elements act like delta functions in momentum variables ("kicks"). Thick elements have input and output at separate physical places, so are more complicated. The thick elements in ETEAPOT are bends, so have potential barriers (hills to climb/fall) as well.

An important normalization for thick elements is that all simulated particles are viewed as traversing the same projected angle (θ) in the design plane (Y=0) upon traversing them ("leaving" the element). θ tends to be called "th" in the code. Typically th is a "split" angle, making the thick element more nearly thin (at the cost of more computation). Calling the actual angle travelled by a given partice ψ, it's projection is θ, and $\psi = \psi(\hat{n}, \theta)$. Diagrams to follow in the "Munoz Applied to the Code" Section.

Phrases like "just inside the output face of the bend element" ([2], page 74) arise in thick elements. Also, "just past the bend entrance" ... The phrase "input face" is synonymous with "entrance". The phrase "output face" is synonymous with "exit".

Clearly it is difficult to specify all these things correctly, and unambiguously. This is more of a qualitative, motivating, look intended to aid subsequent efforts, especially my own. One of the primary purposes of this document is to improve the overall clarity of ("shine light on") this entire process, starting with the next section.

Possibly, also, there is a concept of "local" versus "remote" equations. Local equations only use deviation values. Remote equations use values such as ρ which depend on origin. I tend to use r_D, the design radius, for ρ. Typically, $r_D = 40m$ or so. Typically a "betatron" transverse deflection is 1 cm or so. I'll symbolize this general transverse deflection as "x" for a moment. This is a factor of ca $40/0.01 = 4000$ difference in magnitude. It illustrates why deviation values can typically be viewed as differential/infinitesmal. A best practice observation is that avoiding direct usage of r_D in the equations can save 3 or 4 digits of accuracy.

Really, it is the change in x, $x_{out} - x_{in}$ which is of interest in tracking simulations. An example of how to mitigate the r_D "DC offset" is equation (84) towards the end of Section 4.1 in [2]. Section 4.2 then expands upon this.

One can also distinguish spatial variables from momentum variables. Note that the coordinate systems in Figures 1, 2 have their origins at the element center (zero deviations on design orbit). This makes them "local coordinate systems". Perhaps they should be contrasted with "remote coordinate systems". This is also pursued, starting in the next section.

It may be a bit of an over generalization, but it seems as though "local" and "remote" equations can be massaged into each other with care. It may not be exactly a best coding practice, but some awareness of these issues from the beginning is probably a good idea.

Note that spatial angular momentum depends on origin, while spin angular momentum does not depend on origin. Spatial angular momentum, $\vec{L}$, is even conserved in central force fields. This is the basis for the exact/symplectic tracking in ETEAPOT. More on this to follow.

4 Accelerator Coordinate Systems Overview

This will target electric bends as this is where the "local versus remote" issue arises, and the diagrams are perhaps most elucidating. Choosing the origin of the remote coordinate system at the bend center then makes sense. This was called "centre of curvature" above. Then the out coordinate, $x = r_D + dx_0$, is almost certainly not small relative to r_D!

The theory of [1] is the primary reason for placing the coordinate origin at the bend center. This makes "remote coordinate systems" all but synonymous with "Munoz coordinate systems" here.

Figure 3 is the first diagram in a series intended to gather and illustrate all this. It is intended to depict the $Y = 0$ (design orbit) plane. I identify X with "Out" in it. I hold off on "Up" and "Forward" until the next Figure (4).

This series is intended to motivate the "6D" geometry, and contrast a "local coordinate system" with a "remote coordinate system". Hopefully this sequence will also help to envision the essential "6D" nature of this physics. Figure 3 depicts CW propagation, and is right handed, but the origin is at the bend center. Bend radius is denoted r_D (D for design). r_D may enter the propagation equations.

Figure 4 identifies an "Up" essentially equivalent to Y, though displaced (essentially by r_D along X). Identifying Z or S as "Forward" is a MAD/TEAPOT tradition (with their local coordinate system). There, Z and S (arc length) are substantially synonymous, and the terminology is apt. Here, Forward maybe isn't quite right for the remote reference system because its origin is far from the particle trajectory. Perhaps this is too subjective!

Figure 5 is intended to situate an individual particle trajectory in this coordinate framework. It is thus essentially equivalent to Figures 1, 2. It schematizes a bilateral orbit, though this will be disposed of shortly.

5 Remote Coordinate Development

With the 3D stage set, I now turn to a slightly more quantitative development of electric bend propagation.

Momentum magnitude, and kinetic energy, are not conserved as in magnetic bends.

As mentioned above, angular momentum, $\vec{L}$ is conserved. This can be seen from $\frac{d\vec{L}}{dt} = \frac{d(\vec{r}\times\vec{p})}{dt} = \frac{d\vec{r}}{dt} \times \vec{p} + \vec{r} \times \frac{d\vec{p}}{dt} = \frac{\vec{p}}{\gamma m_p} \times \vec{p} + \vec{r} \times (-\frac{k}{r^3}\vec{r}) = \vec{0} + \vec{0} = \vec{0}$

I will use $\vec{r_{in}}$ here to identify a particle that has just entered a bend, leaving fringe field considerations as already taken care of.

Figure 6 is intended to depict the very specialized case where an exact circular orbit is achieved, but not the design orbit, in the cylindrical geometry ("latitude orbit"). Theory for this type of orbit is in [13]. More general orbits tend to require numerical procedures and simulation.

Figure 7 is intended to depict the very specialized case where an exact circular orbit is achieved, but not the design orbit, in the spherical geometry ("great circle orbit"). Theory for this type of orbit is in [13]. More general orbits probably require numerical procedures and simulation. This geometry is special here, however, because of the "bend, kick, bend" approach of ETEAPOT.

Clearly general initial values must be dealt with. They can be schematized as $(dx_0, dx'_0, dy_0, dy'_0, dt_0, dE_0)$. The code uses these, though scaled. A fairly standard notation is to symbolize these as $(x_0, x'_0, y_0, y'_0, t_0, dE_0)$. In the code, class \$UAL/codes/PAC/src/PAC/Beam/Position.hh, holds these deviations.

Figure 8 is intended to depict the general initial conditions associated with each simulated beam particle (probe), and each bend entrance.

$$\vec{r_{in}} = (r_D + dx_0, dy_0, 0) \tag{1}$$

$$\widehat{r_{in}} = (cos(\phi), sin(\phi), 0) \tag{2}$$

Motion in a central field ($\sim \hat{r}$) is planar because angular momentum ($\vec{L}$) is conserved. Figures 9 and 10 are intended to depict this, generally, and in the Coulomb central field case specifically. The Coulomb central field is special in the code because of the theory of [1]. The normal to the plane, $\hat{n} = -\vec{L_{in}}/L_{in}$, plays a "central" role.

Both Electric field and Energy start with E making their designations clash a little bit.

6 Most Detailed Accelerator Coordinate System Diagrams

I'll start with Figures 11, and 12 (from [2]) as they emphasize the very important angular momentum vector, $\vec{L}$ that I've just been discussing.

There is no real order here, however. The program xfig was used to create the figures mentioned here (in this section). It allows rapid, professional results. The reflection capability is apropos, because it allows simple presentation of both CW and CCW propagation.

Figures 13, 14, 15, 16, and 17 (also from [2]) culminate the accelerator coordinate system overview, and are "state of the art".

The symbolization in the figures in this section is a little different to some of the earlier figures. I leave it to the reader to do his own free association and visualization.

"Small, filled isosceles triangles" for arrow tips seem to look more proper/technical than separate "wings".

7 Equations

As already alluded to, "local" equations are preferrable to "remote" equations. [14], [15] have examples of local equations. A Taylor series for relevant varying quantities such as longitudinal and transverse electric field, to order 5 or so seems to be the consensus. This seems to be more analytically/numerically reliable than the remote expressions of [1] e.g.

$$\vec{F} = q\vec{E} = -k\frac{\hat{r}}{r^2} \tag{3}$$

Here q is the probe electric charge. A consistent symbolization for this for the various particles of interest is probably a good idea. However, the only numerical discrepancy is a factor of minus 1. This "global/remote" expression is essentially "unfamiliar" to an accelerator physicist, used to small deviations from zero for the various relevant quantities. $r = r_D + dx_0$ is not near zero typically!

A more general parameterization is

$$\mathbf{E}(r,0) = -E_0 \frac{r_0^{1+m}}{r^{1+m}} \hat{\mathbf{r}}, \tag{4}$$

and the electric potential $V(r)$, adjusted to vanish at $r = r_0$, is

$$V(r) = -\frac{E_0 r_0}{m}\left(\frac{r_0^m}{r^m} - 1\right). \tag{5}$$

Here, "m", is associated with the "geometry" of the electric bend elements:
m = 0: Cylindrical, "off code geometry"
m = 1: Spherical, "on code geometry"
m $\neq$ 0, 1: "off code geometry"
Now

$$\hat{n} = -\vec{L_{in}}/L_{in} = -\vec{r_{in}} \times \vec{p_{in}}/(r_{in}p_{in}sin(\theta_{r_{in}p_{in}}) \approx \widehat{p_{in}} \times \widehat{r_{in}} \approx \hat{y} \tag{6}$$

because the angle between $\vec{r_{in}}$ and $\vec{p_{in}}$, $\theta_{r_{in}p_{in}}$, is $\approx \pi/2$ ($\vec{r_{in}} \approx (r_D, 0, 0) \approx r_D\hat{x}$, $\vec{p_{in}} \approx (0, 0, p_D) \approx p_D\hat{z}$).

This "feels" like left handed propagation to me as the left hand curled from $\vec{r_{in}}$ to $\vec{p_{in}}$ gives the orientation of $\hat{n}$ as along the thumb.

"On code geometry" refers to the special case the Coulomb field (3) plays in the code. This corresponds to m = 1 in Eq. 4.

8 Munoz Theory

The Hamilton-like vector is

$$\vec{h} = \gamma\vec{v} - \frac{k\gamma}{L}\hat{\boldsymbol{\theta}} = \gamma\dot{r}\hat{\mathbf{r}} + (\gamma r\dot{\theta} - \frac{k\gamma}{L})\hat{\boldsymbol{\theta}} = (\gamma\dot{r}, \gamma r\dot{\theta} - \frac{k\gamma}{L}) = (h_r, h_\theta) \tag{7}$$

by definition. [1] uses it to develop relativistic Keplerian theory.

Symbolically, the following equations are motivated

$$r = \frac{\lambda}{1 + C(Acos(\kappa\theta) + Bsin(\kappa\theta))}, \tag{8}$$

and

$$\frac{dt}{d\theta} = \frac{r}{Lc^2}(k + r\mathcal{E}). \tag{9}$$

I use variables r and θ here, though θ tends also to get used for the design orbit.

9 Munoz Applied to the Code

Figure 19 depicts quantities for the design orbit/particle. θ, or $s = r_D\theta$, is the primary independent variable, the same for all particles, in some sense.

Figure 20 depicts general quantities for an arbitary orbit/particle. It's angular advance in it's bend plane, $\psi = \psi(\hat{n}, \theta)$, allows r_{out} to be evaluated from (10), which I'll write here as

$$r = \frac{\lambda}{1 + C(Acos(\kappa\psi) + Bsin(\kappa\psi))}. \tag{10}$$

10 Code Organization

This first development cycle prototype is currently at version number 1134 (8/5/2014). Now 1137 (8/26/2014).

This "top of the tree" can be seen at

`http://code.google.com/p/ual/source/list`

This version number that comes with a Google Codes download is a useful label. A best practice recommendation is to reference the exact version number used for the results in any documentation and/or reports.

Highly symbolically, general software stages can be represented as $pre-\alpha, \alpha, \beta, \gamma, \delta, ...$ Normally, or often, this is Alpha, Beta, Gamma, ... and γ is named "Release Candidate". There are other nominal stages such as RTM (Release to Manufacturing), GA (General Availability), Production, ...

Here, I view the Release Candidate stage as where I'm, and the software is, at currently. The software is a "candidate" for another cycle. Maybe it will be "released" as such.

In this next tentative cycle, RTM, GA, ... would be strived for.

Perhaps unusually, the development of ETEAPOT led naturally to sequential phases:

"Transverse"

"Longitudinal"

"Spin"

This was kind of happenstance, because the development started with the full 6x6 dimensionality in mind. Scientific application code such as ETEAPOT, perhaps more than other application areas, tends to have a fairly specific target in mind. This target typically doesn't "partition" easily.

The Transverse went straightforwardly with only a smallish "atan2" bug. The Longitudinal had a famous bug that can be cast in terms of the remote coordinate difficulties I've been describing/documenting above. The Spin went straightforwardly, aided by earlier experience, and tight cooperation between Richard and me. There was one medium hitch where we confused "thick elements" (bends) with "thin elements" (multipoles).

What was familiar, to me, was the "get it working" mindset that often seems to accompany serious development for another (supervising) party. The current ETEAPOT code is then often a little illegible. Among other things, I use the

#include

facility ubiquitously. This makes the code hard to read and probably less "reusable". I should probably have subroutines instead (of #includes). Also, I tended not to remove commented out sections from the code. The documentation could be better (always true?).

As mentioned above, my development style was to accomodate on the fly formula development, and my preference is for the #include facility under these circumstances. It's hard to plan and have architectures and interfaces/subroutines when the documentation is a moving target! Often the code would follow the latest formulae within a day or so. Of course, the same name can be used for an eventual subroutine as the #include file. Sometimes the appraisal/evaluation of the "most appropriate" name (but not its actual code) changes. Another moving target. Of course, this is basically just aesthetic.

Again, a powerful architecture (UAL) was inherited. It "handles the sequencing" in essentially linear accelerators. A wrinkle arose that the fringe field associated with spin propagation in electric bends was not handled linearly. Rather than "passEntry" and "passExit" methods bracketing the main, thick, "traverseBend" method, both effects were subsumed into the traverseBend method.

A best practices recommendation is to have some sort of theory and diagram(s) that overview the programmatic lattice sequencing for each element type. This should be in enough detail that tracing individual particle propagation through an element is straightforward. It probably doesn't have to into the full detail of the interface/subroutine signature in all cases. The name of the subroutine is likely sufficient. Debuggers, especially ddd, can help with all this. Thick elements, especially, can have quite complex exact code structure. Object oriented languages, such as the C++ used in UAL, also add complexity, essentially magnify this.

For these reasons, and others, there will probably not be a formal Release for this first cycle. It will likely be viewed as a "prototype" for a possible next cycle where it would be viewed as a cross checking and organizing resource.

Figure 18 illustrates this notion, though with two Prototype cycles. Richard and I are hoping to treat the current code as an "Operational prototype".

Treating vectors and matrices as a unified whole seems to be worthwhile. Files Vectors.h, Matrices.h, and Matrices.cpp are good tools for this.

One of the main cycle one issues is the "user manifest".

Design frequency, f_D, set in file designBeamValues.hh, is one of the first parameters that needs to be determined for the current lattice in any kind of RF work.

The "design beam" must always be set up. Files include
$UAL/examples/ETEAPOT/designBeamValues.hh
$UAL/codes/ETEAPOT_MltTurn/src/ETEAPOT_MltTurn/Integrator/getDesignBeam.h

Precise matching between the theory implementation (actual code) and the theory ([2]) is obviously a good idea. It is a best practice recommendation to achieve this, and "document it" (metadocument?), as well. In practice, this can take the form of comments in the code identifying the section in the theory they are following. This is almost certainly worthwhile.

To first approximation, the recommendation for cycle two is "reduce clutter". The original "client side" orientation of Nikolay Malitsky was to have a minimum number of files in the top directory (e.g. $UAL/examples/ETEAPOT-ManualTune) and the "support" files in subdirectories. Furthermore, the number of files in these subdirectories should be minimal as well. Thus a client side directory might just have 2 files, a Makefile, and the source file. This has not been adhered to at all in this first ETEAPOT development cycle.

Playing into this recommendation is the current budget constraints. With plenty of funding, a little more leisurely "requirements gathering" phase would be nice. Failing this, a "one new directory at a time" approach will probably be adopted.

11 Recipe Notes

Files

$UAL/examples/ETEAPOT/determineTwissExample_m=-1.00
    determineTwiss, transferMatrix example
$UAL/examples/ETEAPOT/determineTwissCleanRunPlot_Protonium
    determineTwiss, transferMatrices example
$UAL/examples/ETEAPOT/runExampleFor_manualTune
$UAL/examples/ETEAPOT/runExampleFor_spinOrbits

are thought of as "recipes" for exercising ETEAPOT functionality. The first three are associated with transverse calculation and are somewhat archival. The last one, runExampleFor_spinOrbits, is likely to be most relevant. It explicitly identifies the sequence of command line commands that can be executed to simulate and view longitudinal and spin propagation.

12 Spin

Simulating/tracking spin propagation and its bunch decoherence time were the main motivation for ETEAPOT.

In ETEAPOT, and probably all approaches, a particles spin does not affect its orbit. The spin "hitch hikes" or "piggybacks" on the orbit.

This can lead to confusion because the dominant orbital lattice elements are the bends, while the dominant spin lattice elements are less certain.

There is reason to believe that a dominant source mechanism of spin decoherence is the fringe field of the bend. This is all but negligible for the orbit.

A significant complication concerns the sign of the $\widetilde{\Delta\alpha}$ ([2], Eq. 326). For a horizontally focusing or defocusing quadrupole there is no ambiguity, since the bend plane in the quadrupole is the same as the overall (horizontal) lattice design plane.

The bend plane is an electric construct, less so a magnetic lattice construct.

13 Bugs

The benchmark reports were based on UAL versions prior to UAL::1130.

14 Conclusions

15 Appendix: Munoz Triad

In this section, to make the algebra easier, I'll denote

$$\hat{r} \equiv \widehat{r_{in}} = (cos(\phi), sin(\phi), 0) \equiv (c, s, 0) \tag{11}$$

and

$$\hat{p} \equiv \widehat{p_{in}} \equiv (d, e, f). \tag{12}$$

Thus, up to a normalization factor, $\hat{n} = -\hat{r} \times \hat{p}$ which I'll write as

$$\hat{n} \sim -(c, s, 0) \times (d, e, f) = -\begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ c & s & 0 \\ d & e & f \end{vmatrix} = (-sf, cf, sd - ce). \tag{13}$$

$\hat{r} \cdot \hat{n} = 0$ by inspection. Similarly

$$\hat{\theta} = \hat{r} \times \hat{n} \sim (c, s, 0) \times (-sf, cf, sd - ce) = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ c & s & 0 \\ -sf & cf & sd - ce \end{vmatrix} = (s^2 d - sce, c^2 e - scd, f). \tag{14}$$

$\hat{r} \cdot \hat{\theta} = 0$ by inspection.

The final dot product is

$$\hat{n} \cdot \hat{\theta} \sim (-sf, cf, sd - ce) \cdot (s^2 d - sce, c^2 e - scd, f) = -s^3 fd + s^2 fce + c^3 fe - c^2 fsd + sdf - cef. \tag{15}$$

This equals

$$s^2 f(ce - sd) + c^2 f(ce - sd) + sdf - cef = 0 \checkmark. \tag{16}$$

16 References

References

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

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

[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

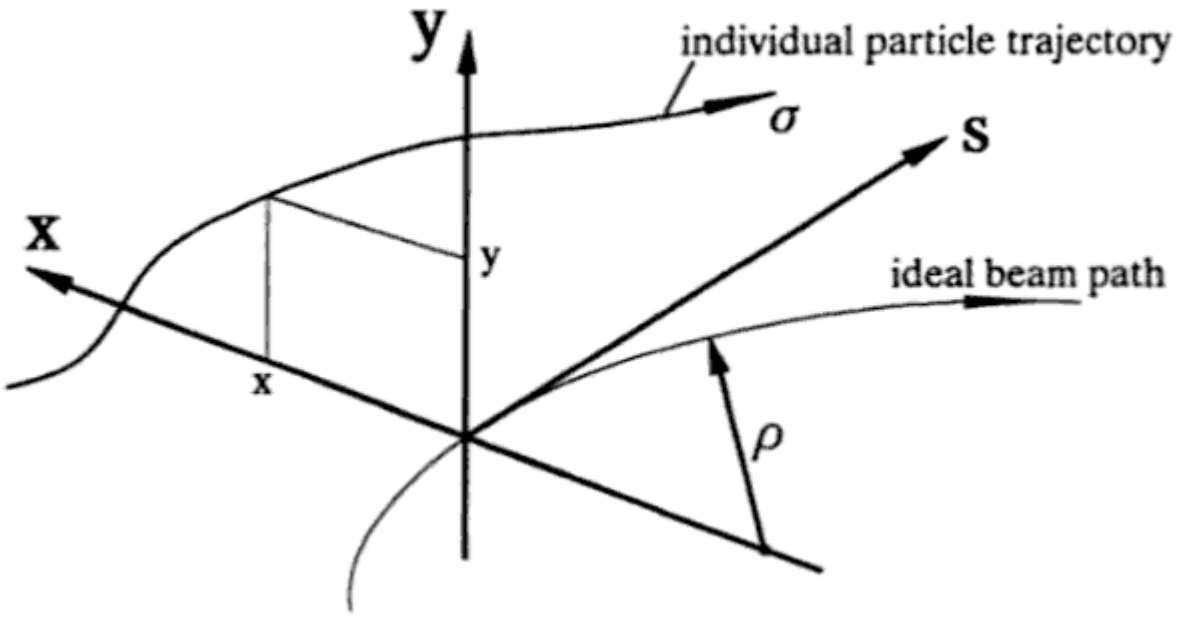

Figure 1: Local Reference System ([3]). Possibly the σ Individual Particle Trajectory Curve Should be "Broken" at the **y** Axis to Emphasize That it is "Behind" it. **x**, and **s** Axes Treated $\approx$ Symmetrically a Plus?

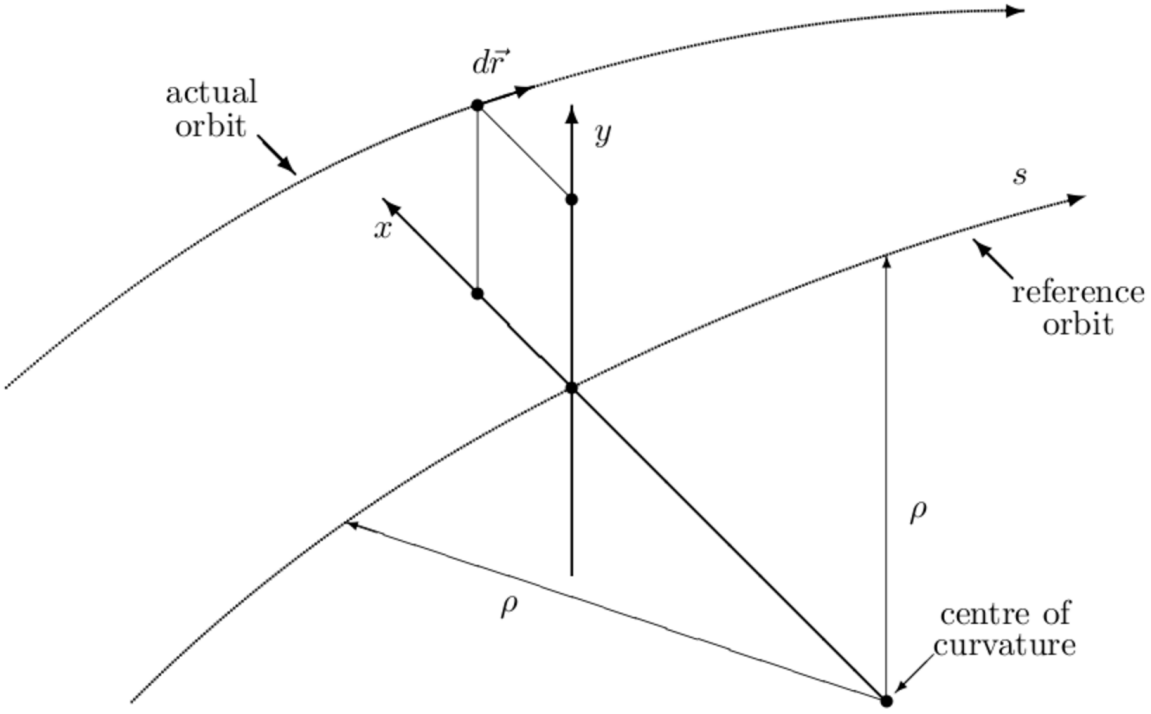

Figure 1.1: Local Reference System

Figure 2: Local Reference System ([5]). The Two ρs May be Slightly Inconsistent? Perhaps it's not Clear That the y Axis is Intended to be out of the Page?

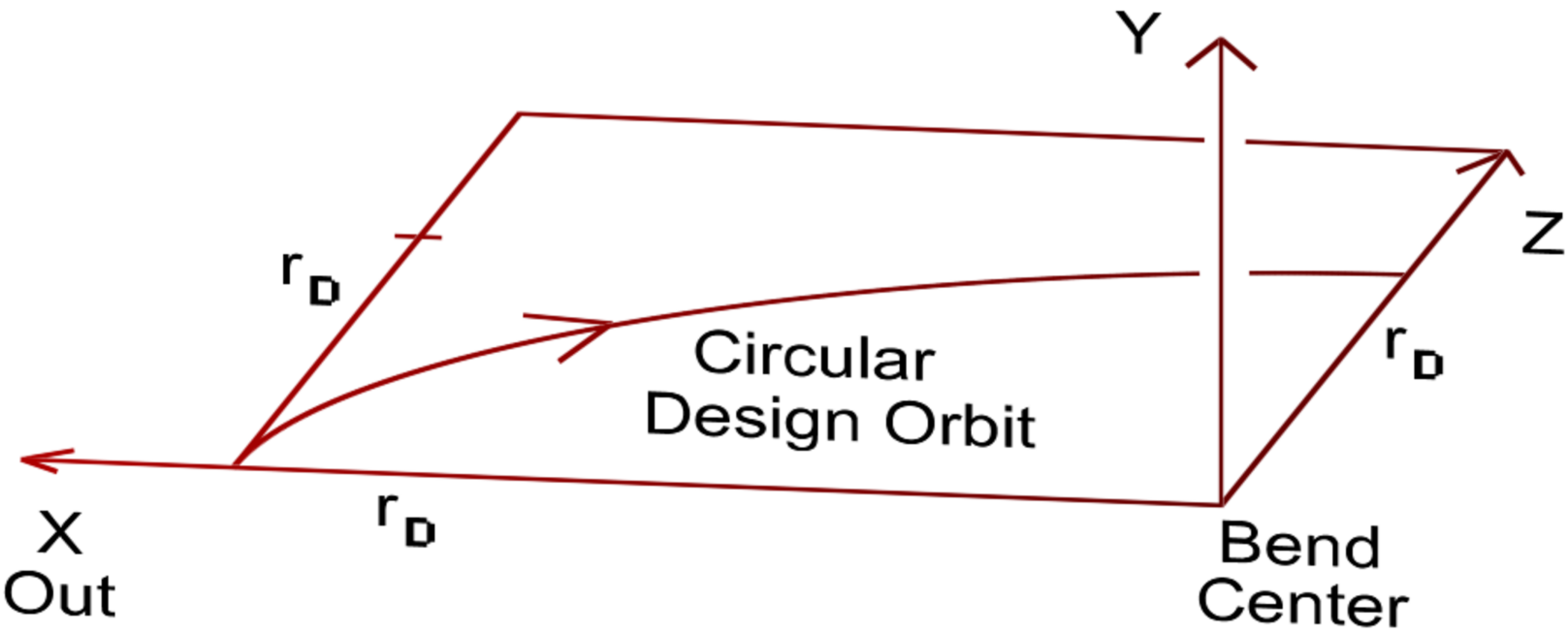

Figure 3: Design Plane (Y = 0) Orientation. A Known, Well Defined, Design/Ideal Particle Reference Orbit is Central to the Whole Paradigm. Of Course, this is not Just Spatial. Actual Particle Deviations from the Reference Particle are What is Maintained (Tracked). In the sxf files used in ETEAPOT, the Design Orbit in Bend Elements is a Circular Arc.

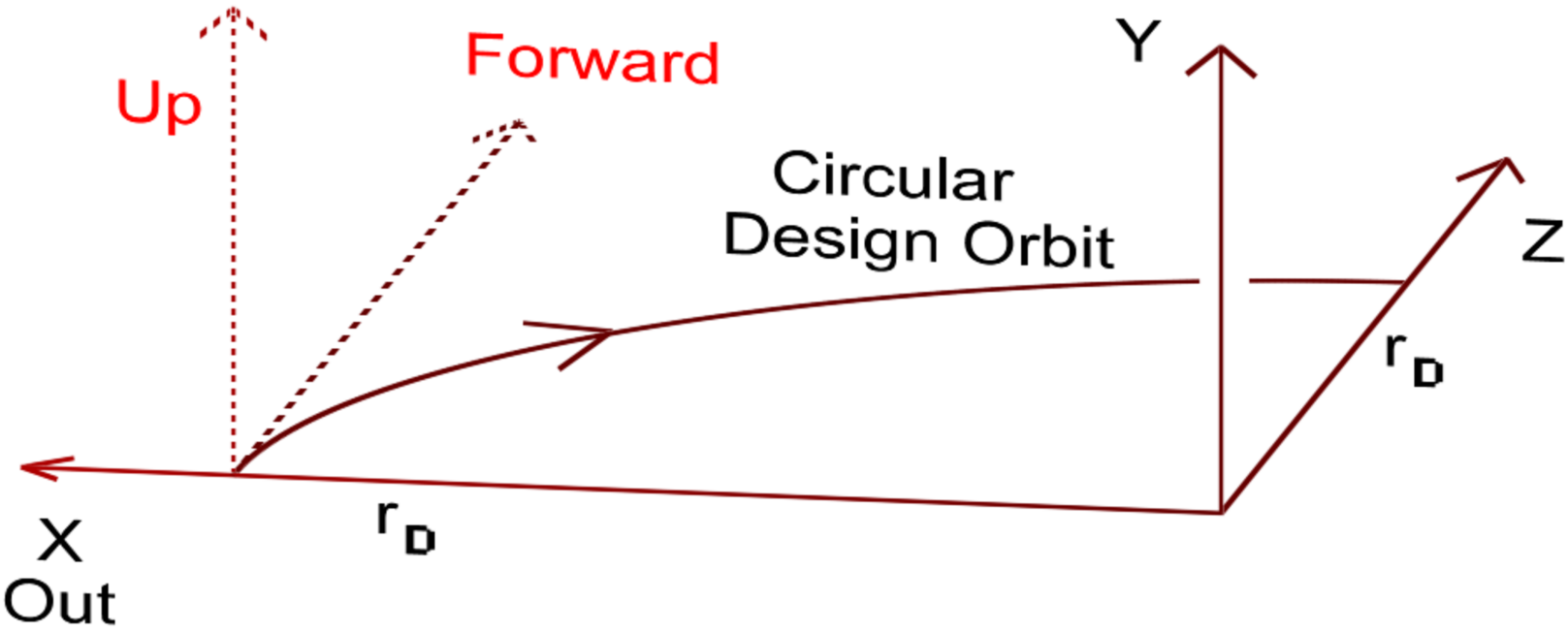

Figure 4: Local Reference System Versus "Remote" Reference System. The Local Reference System is Much More Common.

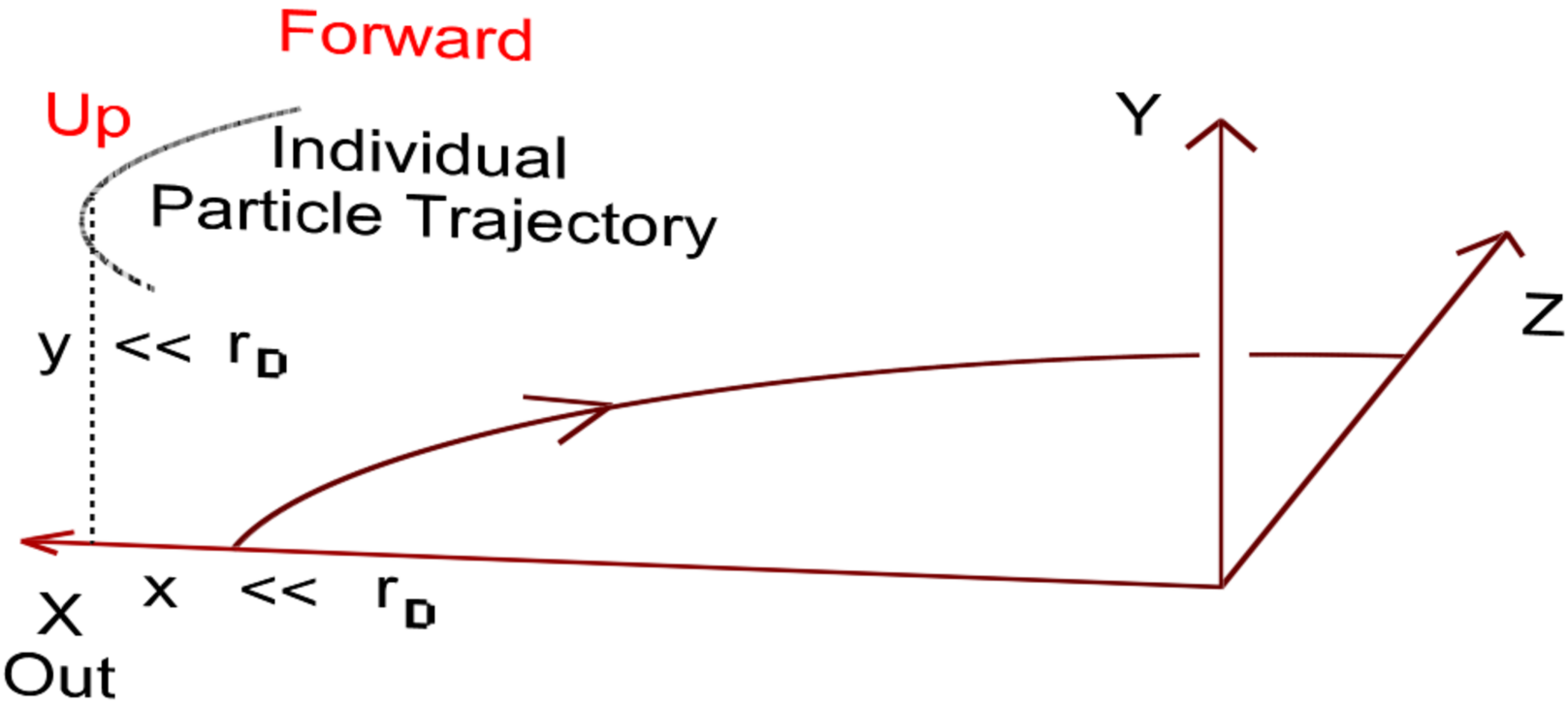

Figure 5: Individual particle (remote) coordinates near bend entrance. The variable x should probably be symbolized dx_0 here (0 for initial, d for deviation). The shorthand used here is typical in the accelerator world. This particle coordinate has several interpretations (Table 1 in [2]), not all strictly Cartesian. Unusually, the quantity $r = r_D + x$ (remote origin) is meaningful in the Munoz theory and code implementation. Best practices suggest combining two quantities differing in magnitude by a factor of 1000+ is a bad idea. Best practices also suggests consistent naming conventions (not always easily practical). A "captured" particle will have an orbit quite close to the circular design orbit. This is obscured here, as this diagram is not to scale. With a line charge along Y, a "latitude" orbit, such as begun here, is feasible. With a central field pointing to the origin, it is not.

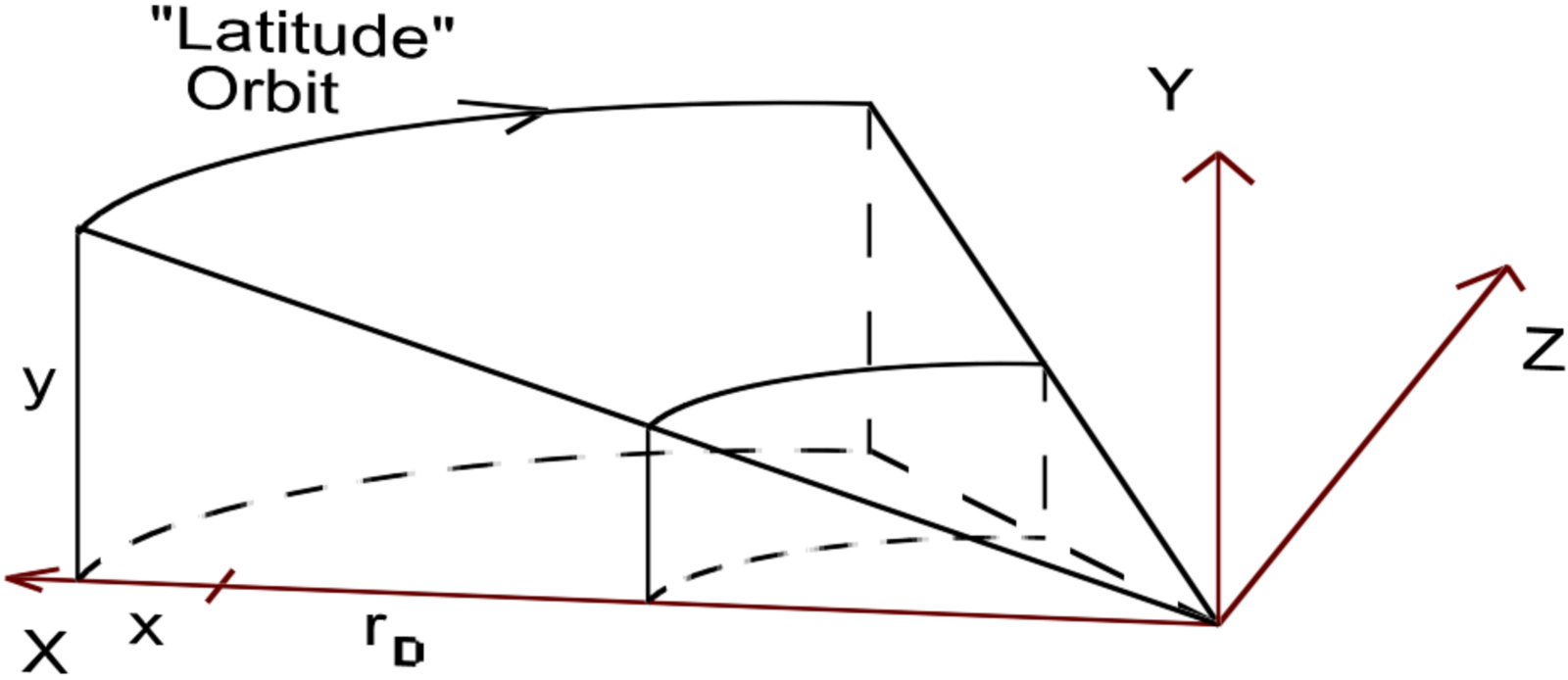

Figure 6: Semi quantitative highly specific (”latitude”) individual particle trajectory. Theory exists for this particular type of orbit. It is both an off momentum circle orbit, and an ”off code geometry” (cylindrical) orbit. The ”on code geometry” is spherical Of course this is bending element terminology. See Figure 7. Clearly this orbit involves very specific initial values in very specific bending geometry (cylindrical).

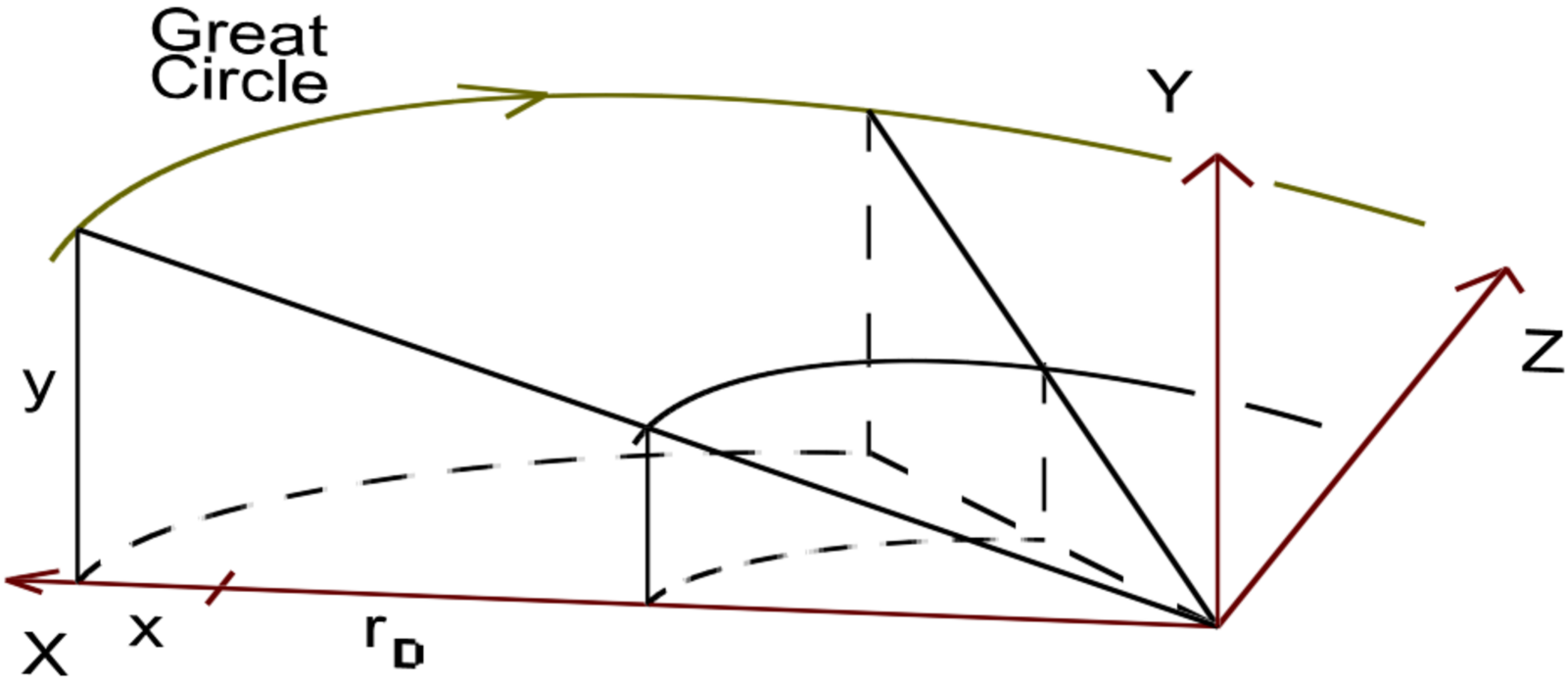

Figure 7: Semi Quantitative Highly Specific ("Great Circle") Individual Particle Trajectory. Note that it's in Green. Orbits in this on Code (Spherical) Geometry are Calculated "Exactly", then "Kick Corrected" so that the Code can Handle General Geometries. Theory Exists for this Particular Type of Orbit (Off Momentum Circle). Spherical Bending Geometry is Essential. Clearly this Orbit Involves very Specific Initial Values in very Specific Bending Geometry (Spherical). The Code Handles General Initial Values, However. Initial Values can be Schematized as $(dx_0, dx'_0, dy_0, dy'_0, dt_0, dE_0)$, Though they are Often Symbolized less Descriptively.

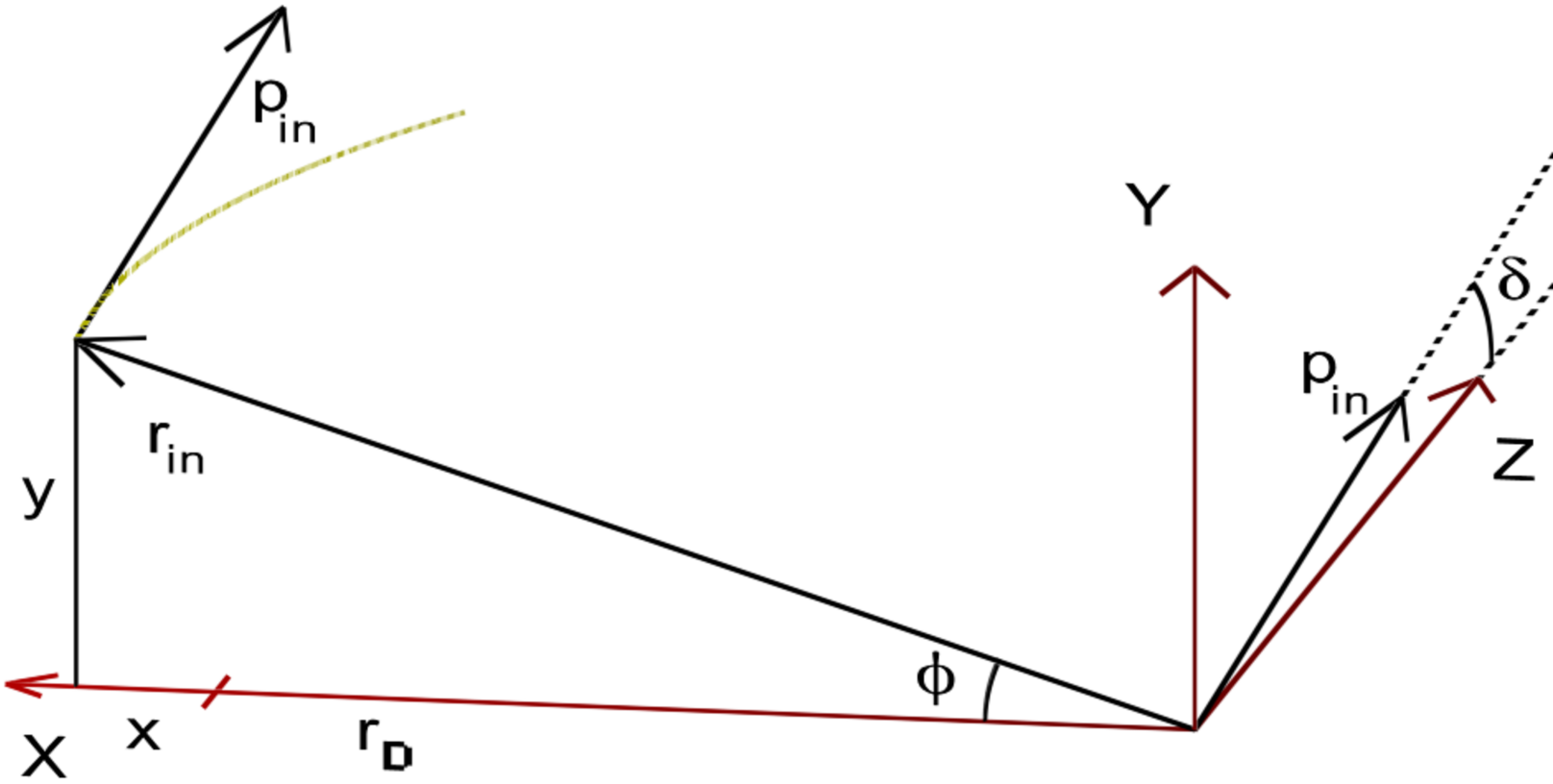

Figure 8: The "code view" of general initial conditions in a remote coordinate system. The symbols here indicate that this is an electric bend segment, though this is not absolutely necessary. There doesn't seem to be much call for remote coordinate systems for other lattice elements (e.g. multipoles). Hopefully this particle will be captured by these optics, and the overall focussing of the lattice. Note that this assumes this probe, $\vec{r_{in}}$ (and hence ϕ) is in the $Z = 0$ plane ($\vec{r_{in}} = (x_0, y_0, 0)$). Time of flight deviation from design particle is maintained for each probe to allow this. $\vec{p_{in}}$ is very close to $p_D\hat{z}$, but can have all nonzero components $\vec{p_{in}} \equiv (p_{x_0}, p_{y_0}, p_{z_0}) \approx (0, 0, p_D)$. Thus δ is not necessarily in the $X = 0$ plane. Orbits (green) in this on code (Spherical) geometry are calculated "exactly", then "kick corrected" so that the code can handle general geometries. This idealized orbit is in the "bend plane" defined by $\vec{L_{in}} = \vec{r}_{in} \times \vec{p}_{in}$. [2], Eq. 120 on page 34, gives the expression for $\hat{n} = -\vec{L_{in}}/L_{in}$ which I tend to work with. $\hat{n}$ is displayed in Figure 9. It indicates left handed clockwise rotation in some sense. Perhaps $\hat{n}$ should be denoted $\hat{n_{in}}$, but $\hat{n}$ is fixed for the bend propagation. Note that this contrasts with $\vec{r_{in}}$ going to $\vec{r_{out}}$, and $\vec{p_{in}}$ going to $\vec{p_{out}}$.

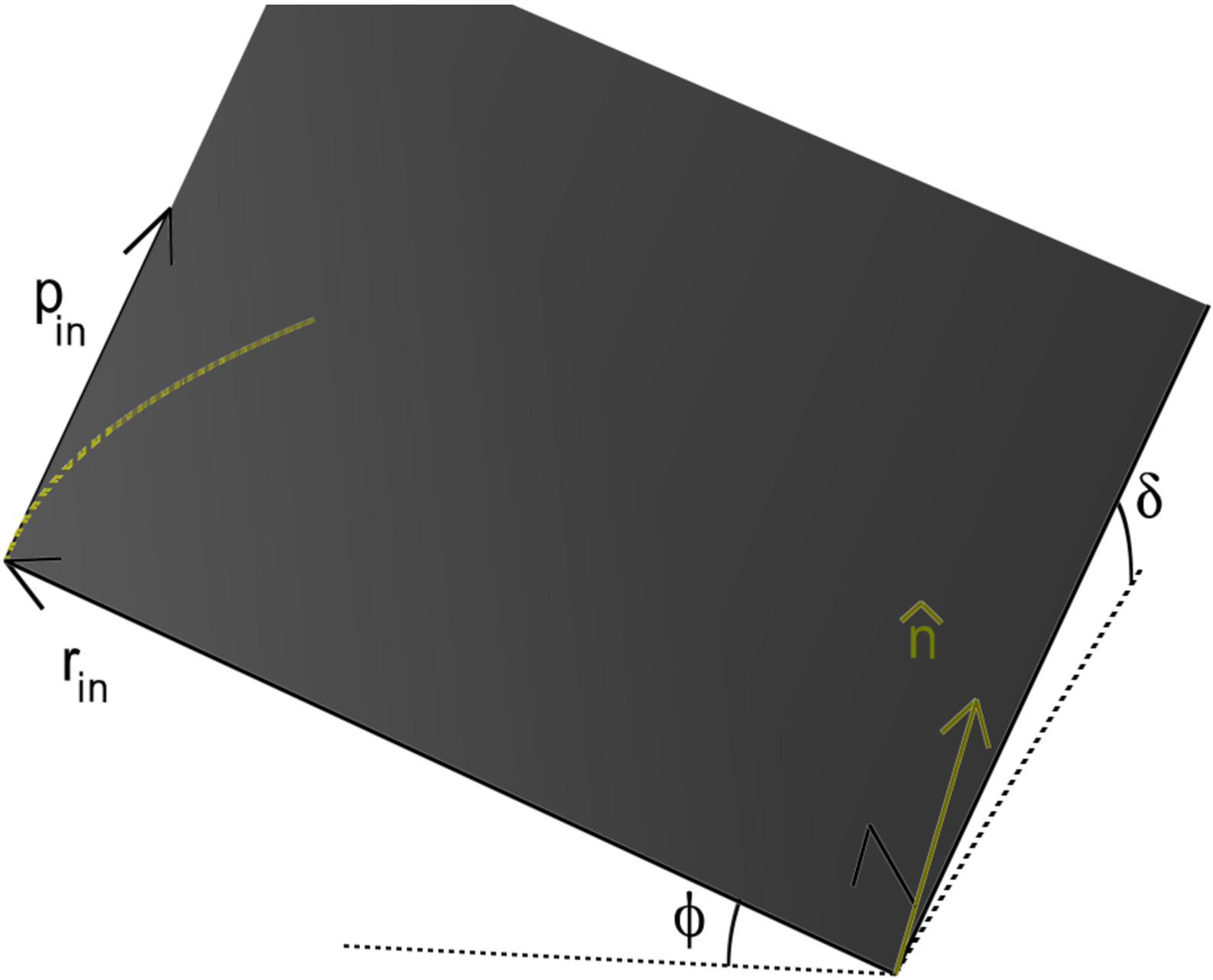

Figure 9: "The bend plane". Its normal, $\hat{n} = -\vec{L_{in}}/L_{in}$, is emphasized. Again, it indicates left handed clockwise rotation in some sense.

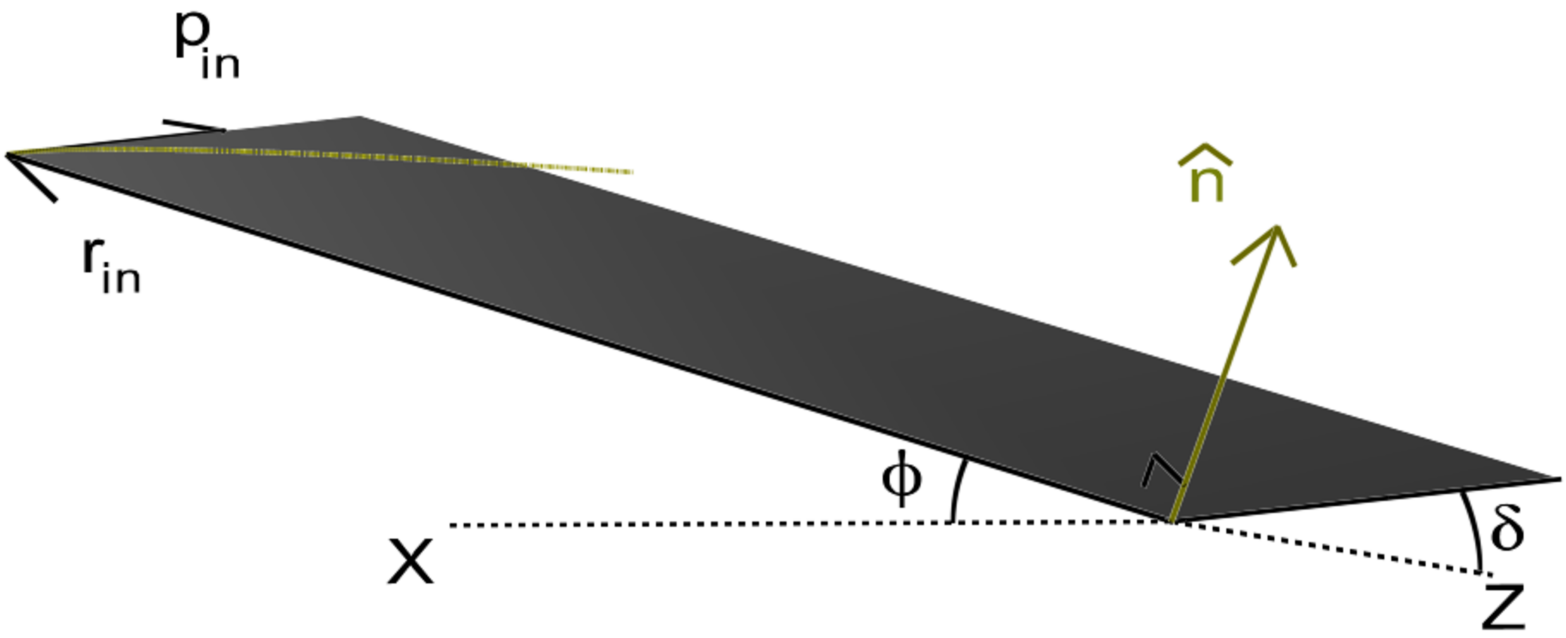

Figure 10: Different slant on the bend plane. Its normal, $\hat{n} = -\vec{L_{in}}/L_{in}$, may be more readily visualizable. Realistic bend planes corresponding to captured particles have ϕ and δ very close to zero. "Paraxial" is one adjective. "Tipped" plane is another, where the tipped is taken to mean very slightly tipped. Again, a typical tip is 1cm/40m = 0.00025 radians.

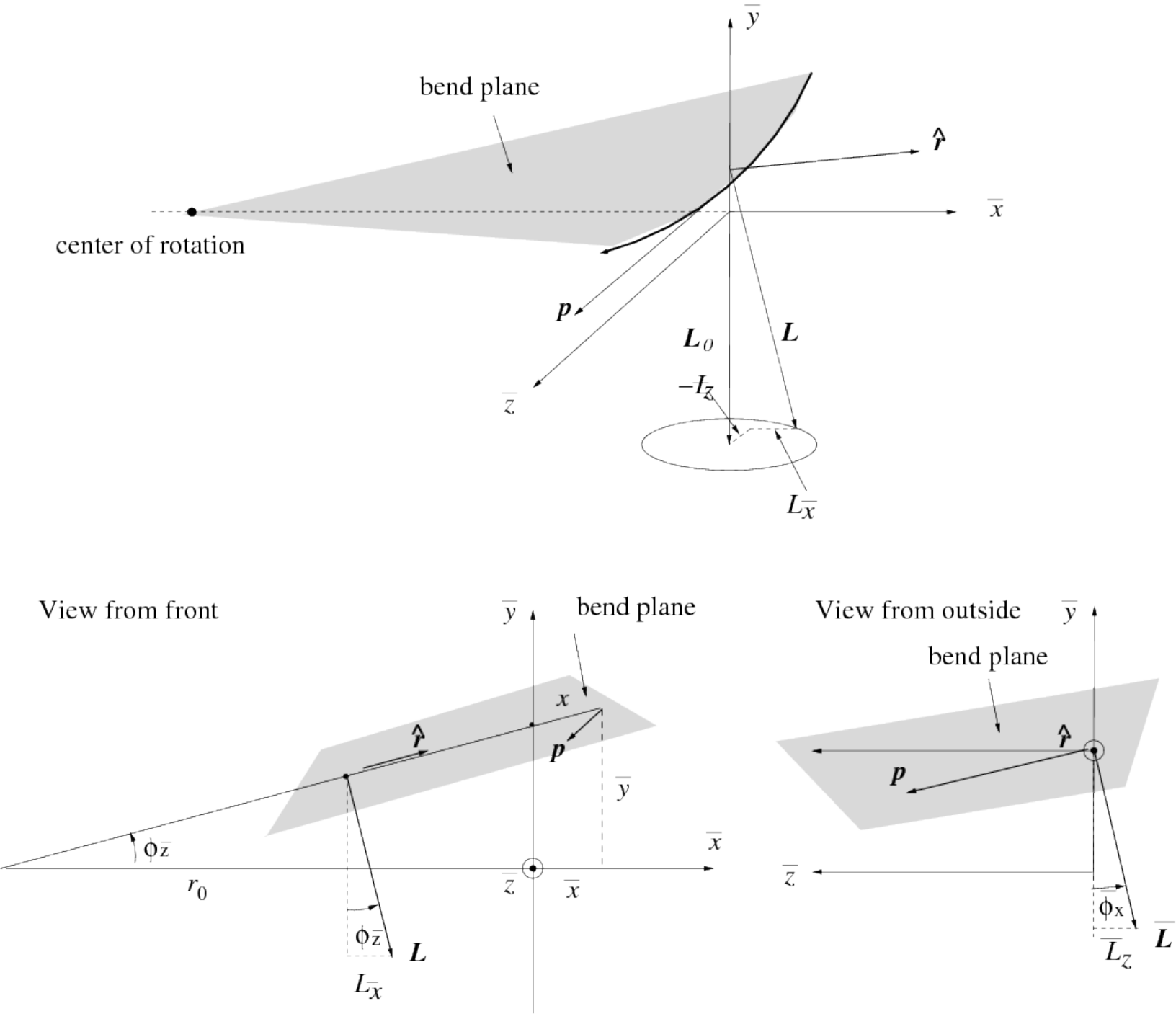

Figure 11: Example "most detailed diagram" (from [2]) emphasizing the very important angular momentum vector, $\vec{L}$.

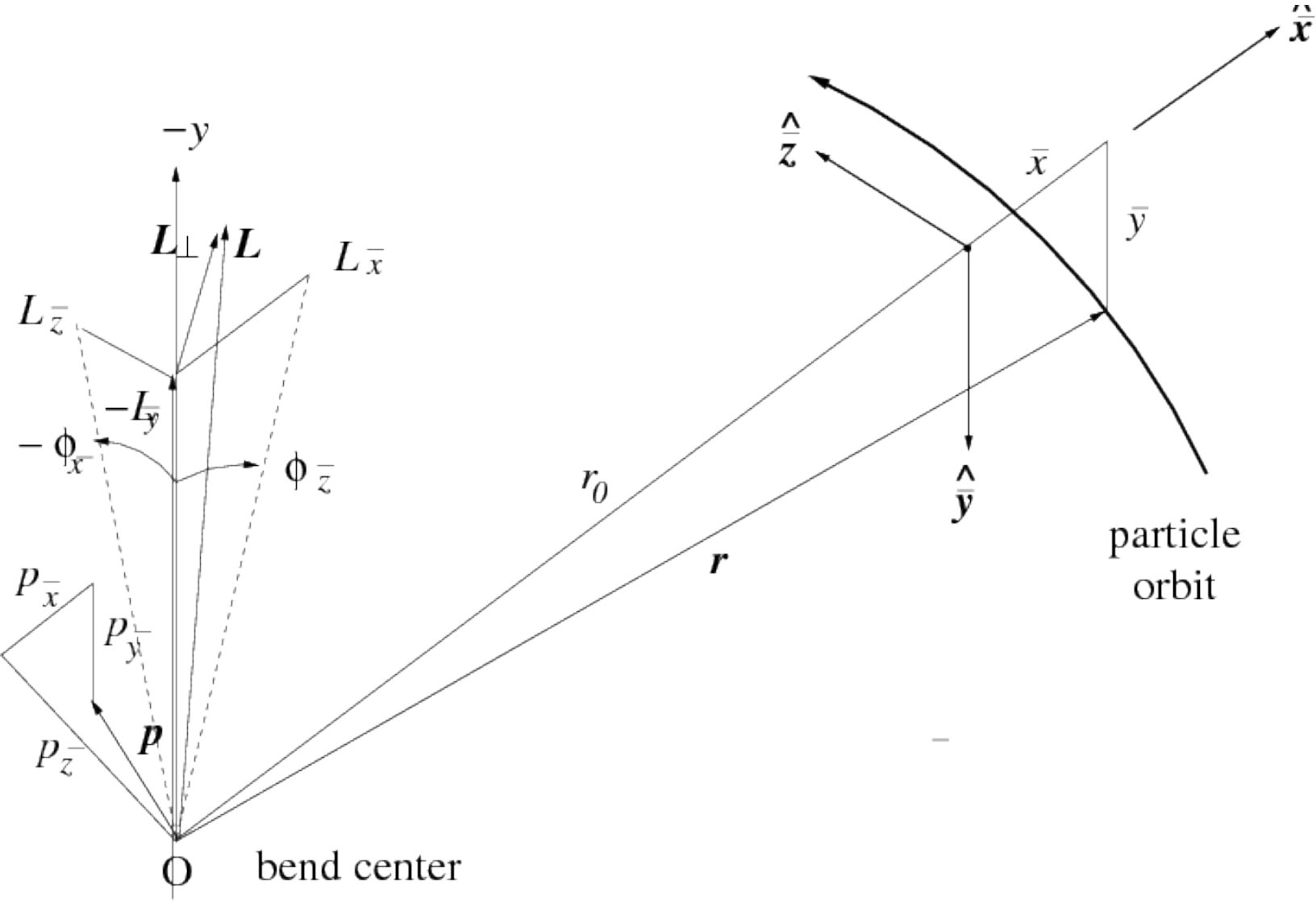

Figure 12: Example "most detailed diagram" (from [2]) emphasizing the very important angular momentum vector, $\vec{L}$.

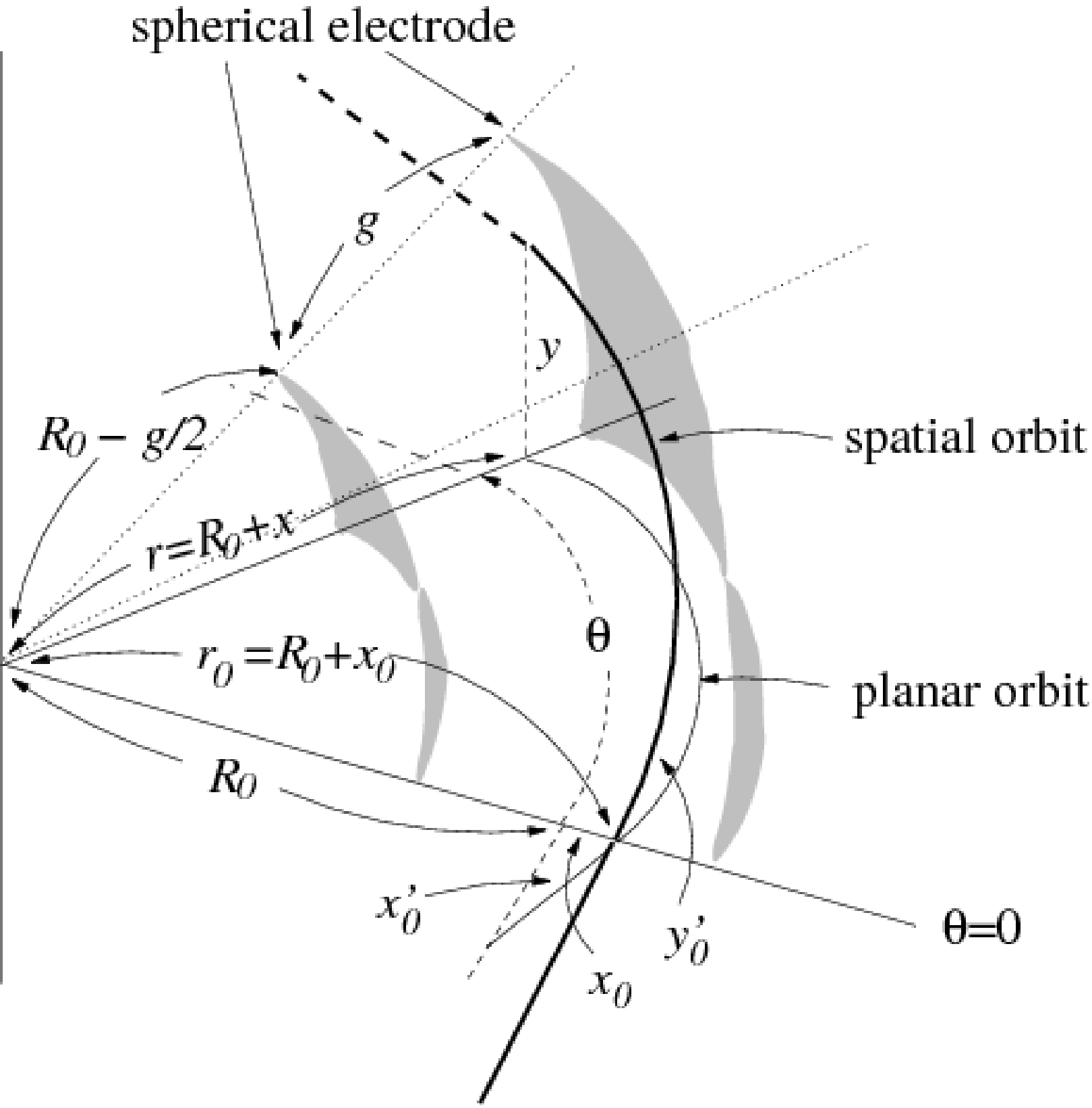

Figure 13: Example ″most detailed diagram″ (from [2])

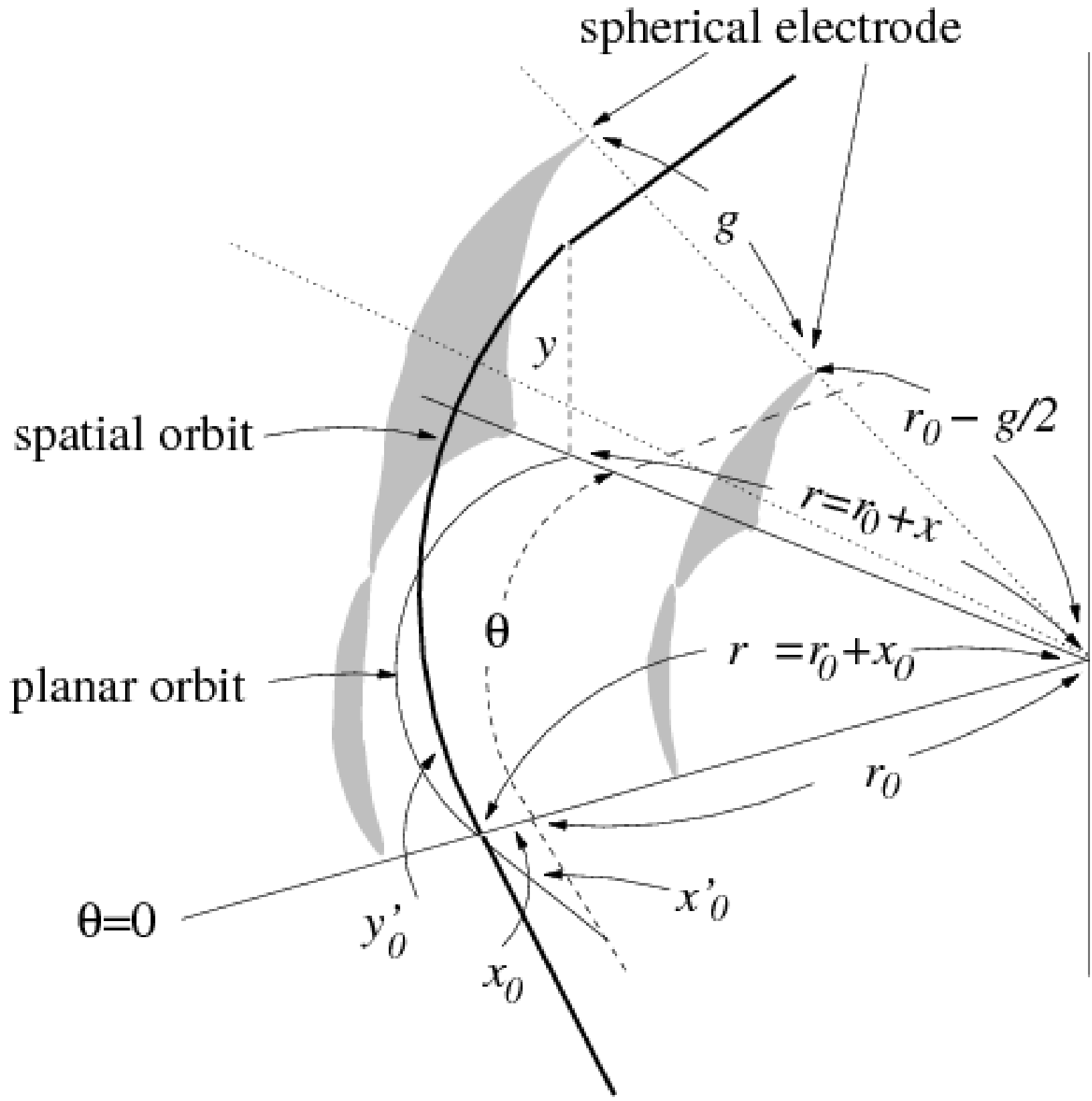

Figure 14: Example ″most detailed diagram″ (from [2])

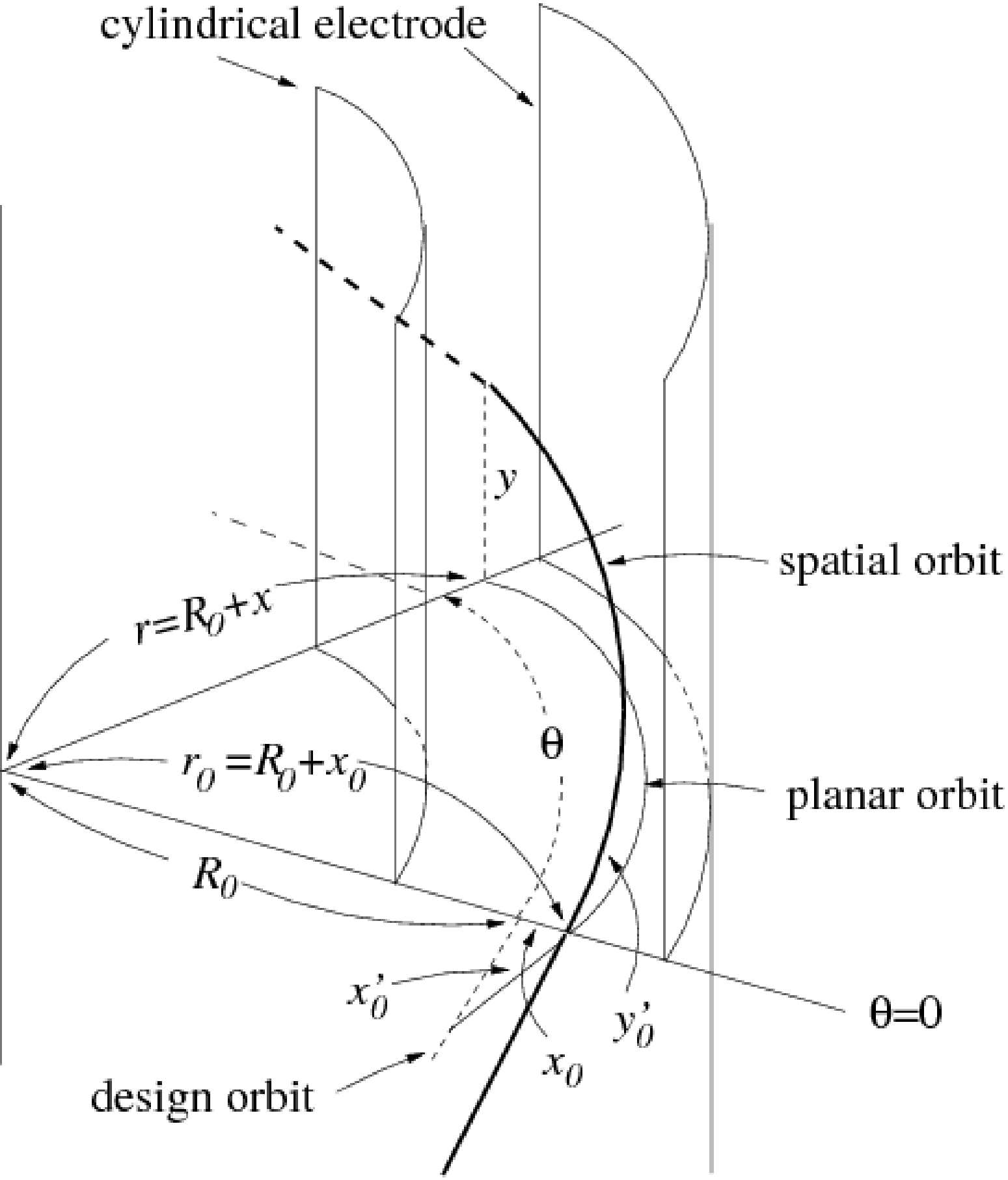

Figure 15: Example "most detailed diagram" (from [2])

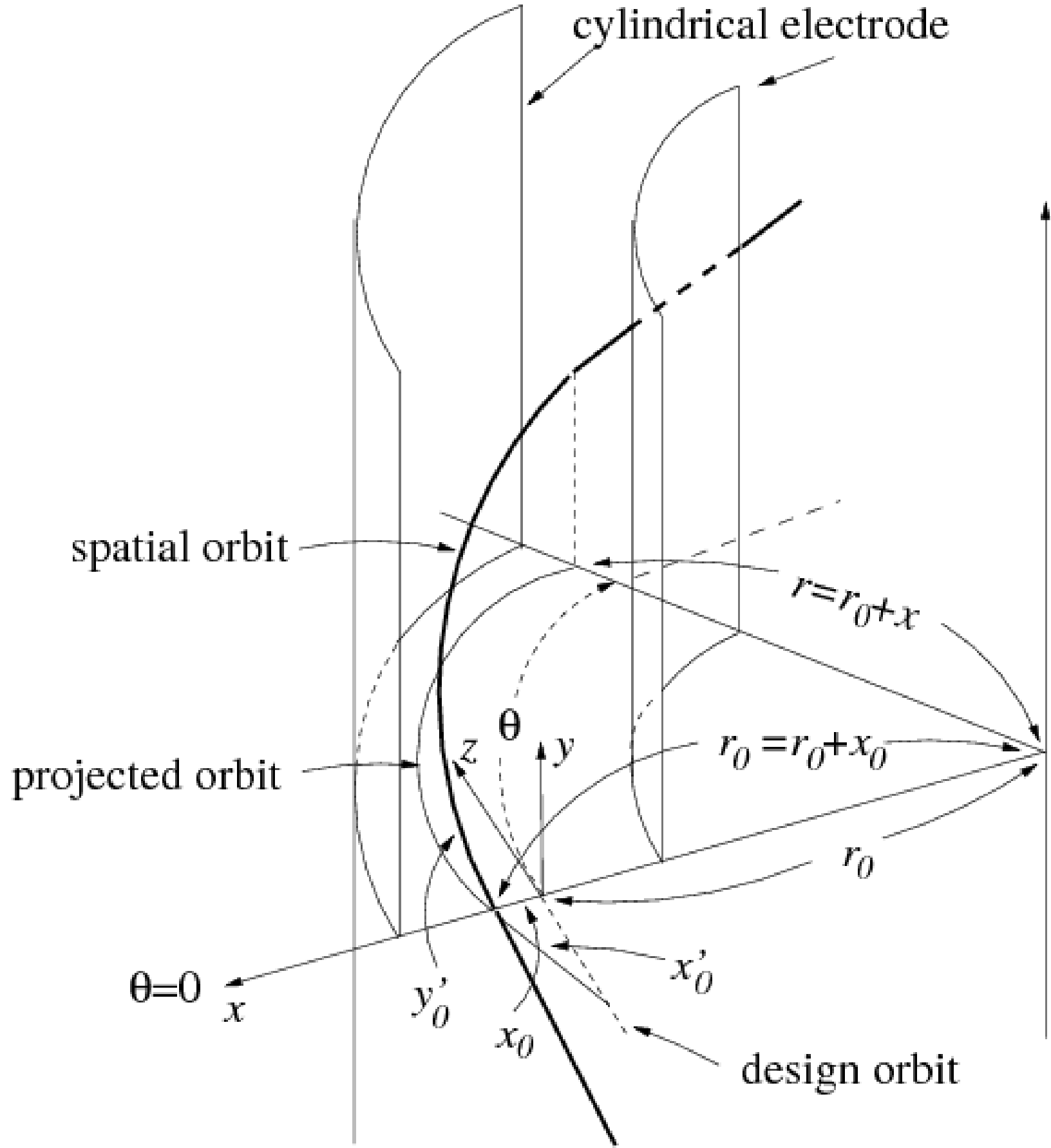

Figure 16: Example "most detailed diagram" (from [2])

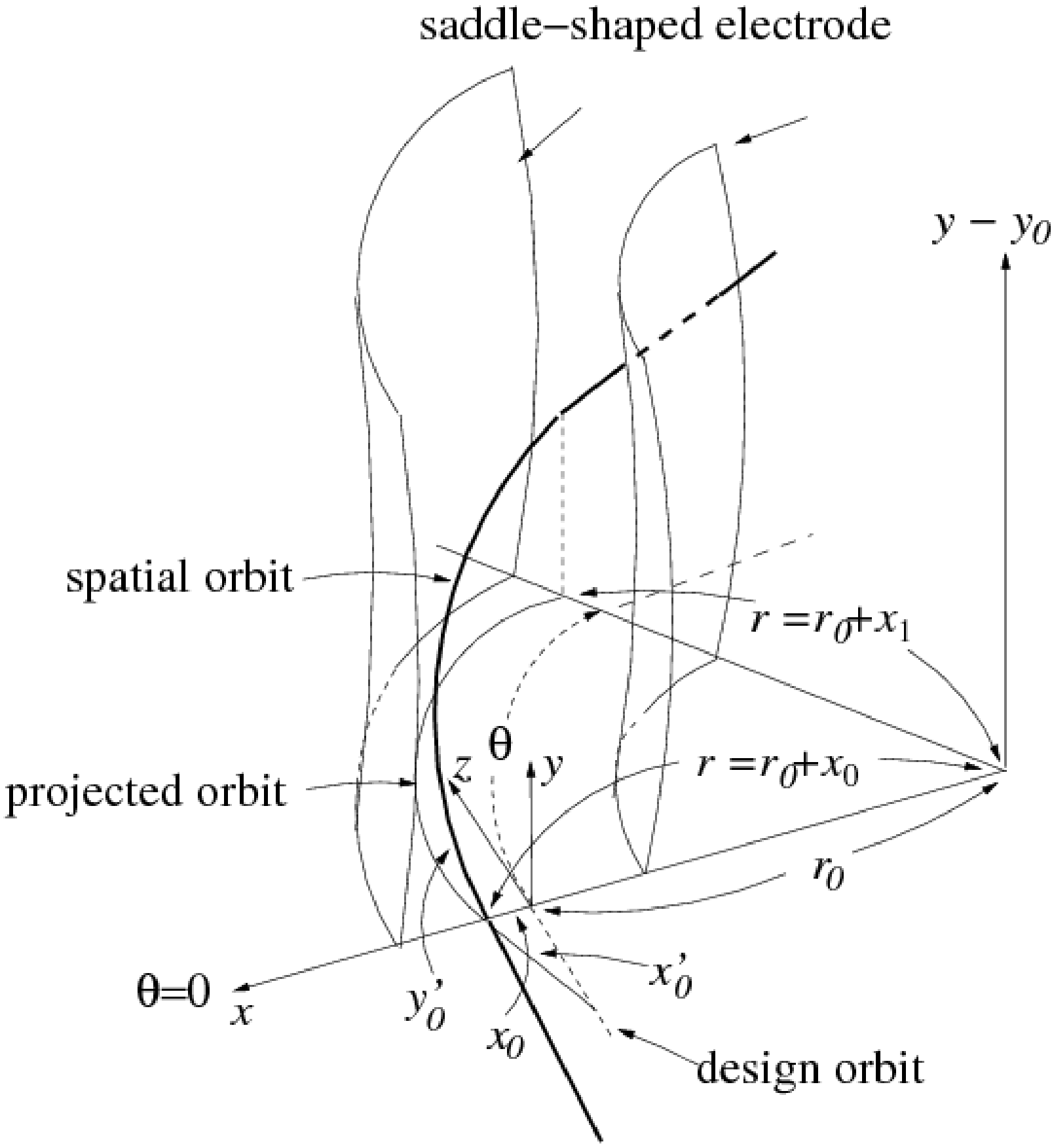

Figure 17: Example ”most detailed diagram” (from [2])

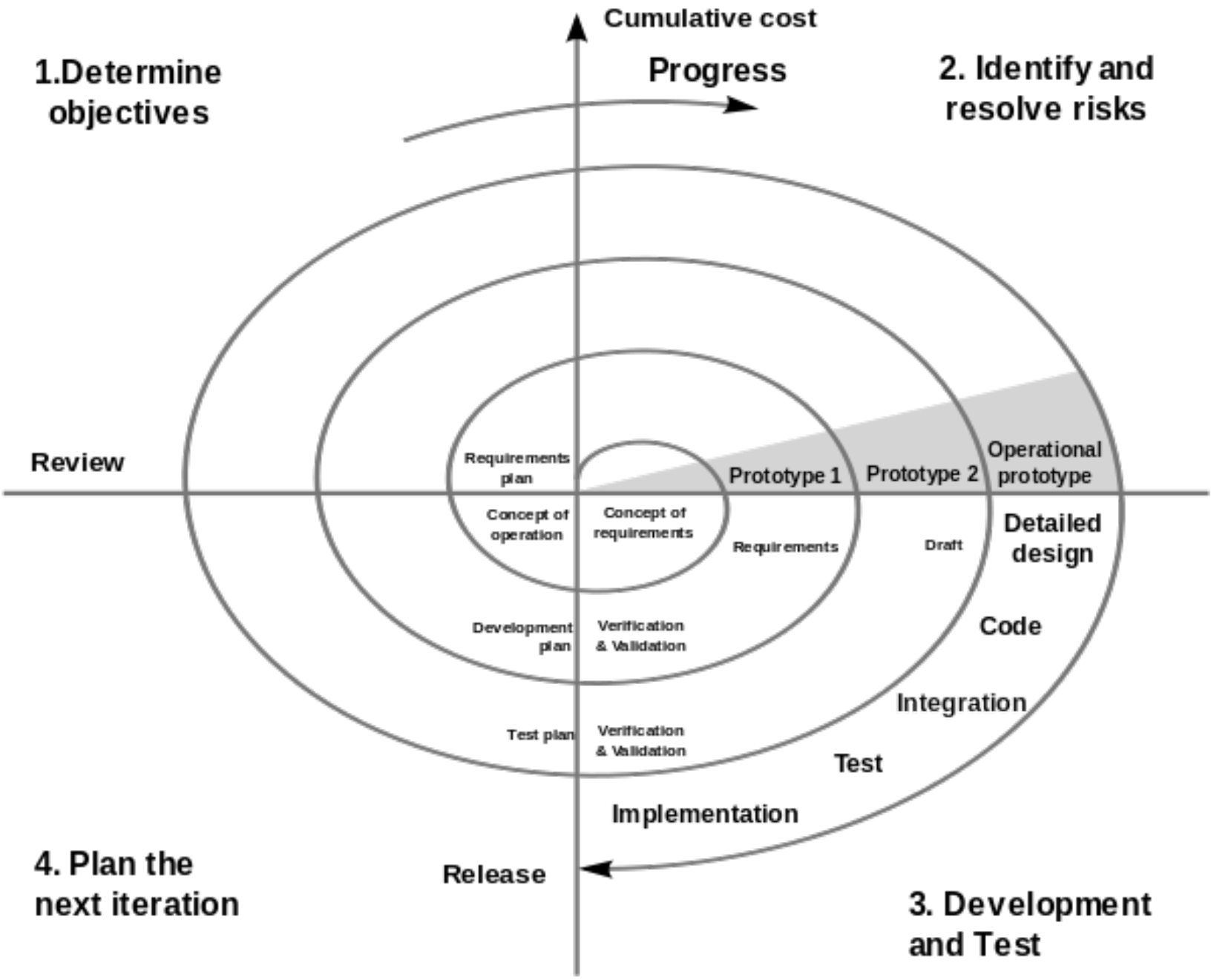
Cumulative cost
Progress
1.Determine objectives
2. Identify and resolve risks
Review
Requirements plan
Prototype 1
Prototype 2
Operational prototype
Concept of operation
Concept of requirements
Requirements
Draft
Detailed design
Development plan
Verification & Validation
Code
Integration
Test plan
Verification & Validation
Test
Implementation
4. Plan the next iteration
Release
3. Development and Test

Figure 18: [17])

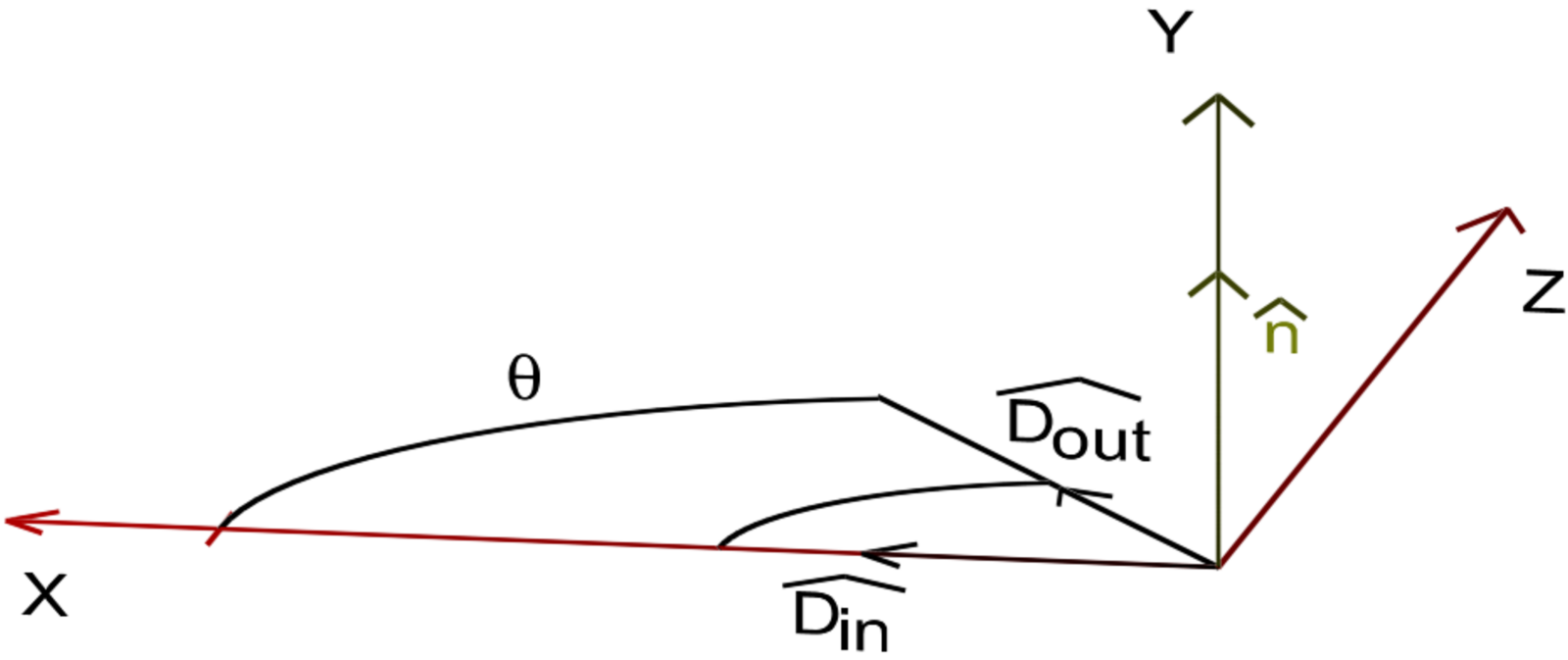

Figure 19: $cos(\theta) = \widehat{D_{in}} \cdot \widehat{D_{out}}$. D stands for Design. θ is the angle to split bend exit for the design particle. It is also the projected angle for all tracked particles. See Figure 20.

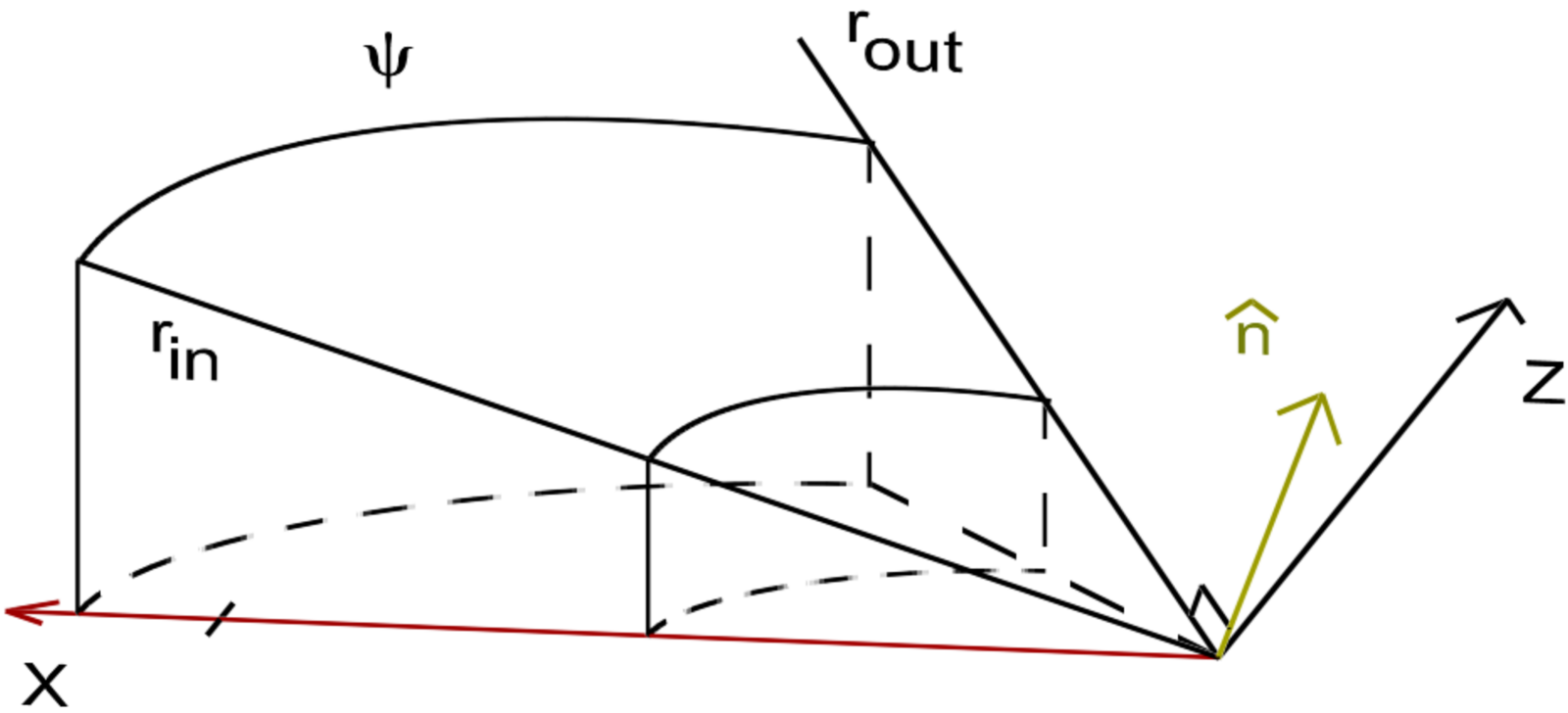

Figure 20: $cos(\psi) = \widehat{r_{in}} \cdot \widehat{r_{out}}$. $\widehat{r_{in}}$ is known from tracking. θ is the projected angle (Figure 19), and is the same for all tracked particles. $\widehat{r_{out}} = a\widehat{D_{out}} + b\hat{y}$. $\widehat{D_{out}} = (cos(\theta), 0, sin(\theta))$. Then a and b can be solved for to give ψ.