# Time integration in the code Zgoubi and external usage of PTC's structures

E. Forest, F. Méot

# Time Integration in the Code Zgoubi and External Usage of PTC's Structures

Étienne Forest
KEK
and
F. Méot
CEA

June 6, 2006

**Abstract**

The purpose of this note is to describe Zgoubi's integrator [1] and to describe some pitfals for time based integration when used in accelerators. We show why the convergence rate of an integrator can be affected by an improper treatment at the boundary when time is used as the integration variable. We also point out how the code PTC [2] can be used as a container by other tracking engine. This work is not completed as far as incorporation of Zgoubi is concerned.

# Contents

# 1 Introduction

So what can we say about Zgoubi?

Firstly, it is an integrator just as PTC is also an integrator. There are no "transport" matrices in Zgoubi.

Secondly it uses time as the indepedent variable. If one reads carelessly the Zgoubi related literature, especially a person with an accelerator background, one may rush to the conclusion that Zgoubi uses a variable "$s$" representing some distance along a fictitious reference orbit of some sort as is common in accelerators. This is absolutely false. Zgoubi uses the time $t$ which is equivalent to the *true* path length denoted by "$s$" in the Zgoubi literature. In this note we will purposely use the word time to make absolutely clear that Zgoubi is integrating the usual time-based Lorentz equations.

The fact that Zgoubi uses time is an extremely important feature. As a result Zgoubi's dynamics is global. PTC or TEAPOT [3] dynamics are local. In PTC or TEAPOT it is possible, depending how one writes the equation of motions, to have a particle exceed the speed of light! This is not a reflection of a bug or even a lack of exactness to use Talman's terminology or an abnormal addiction to science fiction literature, it is a result of what happens when a particle reverses direction in a magnet and goes back up stream in a code which *does not allow it by construction*. It is indeed perfectly acceptable for a charge particle to reverse direction in a dipole, for example trapped ions do it all the time.

It should be clear therefore that Zgoubi can potentially describe many more situations than the standard particle accelerator codes, be they integrators or matrix code. And not surprisingly, Zgoubi is the most complete accelerator code we have ever seen in terms of standard "canned "models.

Thirdly, and this is really mostly a programmer issue, Zgoubi is written in Fortran77 and typifies "algorithmic programming." In fact, most old codes still have a concept of magnets, beam lines and commands which internally are stored in different arrays or structures. In pure old Fortran codes, the magnet data will be sprinkled over common blocks, the command data over a different of blocks, and so on. In Zgoubi the main array is a linear description of the input file of Zgoubi. The input file of Zgoubi contains a linear description of the lattice with commands sprinkled in between. Thus Zgoubi internally cycles around the input file commands! It is simply from a conceptual stand point a giant algorithm. Of course so is any program even if written is pure object oriented language when we look at the assembler level, however here it is patently so at the level of the user and the programmer. This means that Zgoubi has all the advantages of Fortran77 and all its disadvantages as well.

We now examine the special care one must exercise when using time integration in the context of a particle accelerator code. Later we will describe briefly how Zgoubi could use PTC's structures.

# 2 Zgoubi's Equations

Zgoubi plainly integrates Lorentz equation which is simply:

$$\frac{d\vec{p}}{dt} = e\,\vec{v}\times\vec{B} \tag{1}$$

Denoting the path length on the trajectory by $s$, the equation can be rewritten as:

$$\frac{d\vec{p}}{ds} = e\,\frac{\vec{v}}{v}\times\vec{B} \tag{2}$$

Since $v$ is constant, we can extract the relativistic factor $\gamma$ from Equation (2):

$$\frac{d\vec{w}}{ds} = \frac{e}{m\gamma v}\,\vec{w}\times\vec{B} \quad \text{here } \vec{w}=\frac{\vec{v}}{v} \tag{3}$$

We can introduce the familiar relative momentum deviation $\delta$ in this equation:

$$\begin{aligned} \frac{d\vec{w}}{ds} &= \frac{1}{1+\delta}\,\vec{w}\times\vec{b} \\ \text{here } \vec{b} &= \frac{\vec{B}}{p_0/e} \\ \text{and } p &= m\gamma v = p_0\,(1+\delta) \end{aligned} \tag{4}$$

At this point we must restate the main difference between integrator codes such as PTC or TEAPOT. Equation (4) is global: we have lost nothing from the original Lorenz equation (1). In fact, the only type of particles which cannot be described easily is a stationary particle for which $\delta = -1$. Besides that trivial case, all conceivable trajectories can be described by Equation (4). By choosing to use the variable $s$, we have simply assigned a different measurement of time to particles with different energies. In every day life we may ask a visitor to our house "how far down the road do you live?" and, of course, we simply are trying to find out an estimate of the time taken to reach our house. Zboubi rephrases the equation of motion in terms of distances rather than real time.

In contrast, other codes, such as PTC, use a local distance along the magnet to describe the motion. That representation can be as exact as global time integration. However it excludes trajectories which reverse direction in a magnet. In optical systems it is quite natural to adopt this point of view. The entire theory of Courant-Snyder and all their nonlinear generalizations are based on maps from surfaces of section defined **not** at a constant time but at some constant position.

In addition, the symplectic condition which seems so important to preserve in a periodic accelerator is not that of the original time based map but that of the map defined on some surface cutting through the beam pipe. The mathematicians assure us, since Gaston Darboux, that there exist some coordinates on that surface for which the map is symplectic (Hamiltonian) as defined as preserving the usual Poisson brackets. Accelerator physicists, especially people writing codes such as PTC, must go further and actually construct a local Hamiltonian describing propagation between these surfaces. This type of integration, particularly symplectic integration, has become an art.

Zgoubi bypasses this optical or Darbousian description by using time. It does it at a certain price. We will now describe the pitfalls of time integration and how one must be careful when applying it to an optical system with discontineous surfaces. We will refer to systems where local projected coordinates are desired as Darbousian in honour of Darboux.

# 3 Time integration in a Darbousian system

We first recall what is meant by a $k^{th}$ order integrator.

## 3.1 Definition of a $k^{th}$ order integrator

Let us imagine that we have the coordinate at the entrance of a straight element which extends from $z = 0$ to $z = L$ at a time $s = s_0$. We purposely refer to $s$ as the time. It will not change the fundamentals of our discussion except that for an electric element, which does not preserve the modulus $v$, some modifications would be necessary.

The coordinates of this particle are in the redundant extended phase space:

$$\begin{aligned} \vec{\omega}_0 &= (\vec{u}, s_0, \delta) \\ &= (\underbrace{x, w_1, y, w_2, 0, w_3}_{\vec{u}}, s_0, \delta) \end{aligned} \tag{5}$$

We now proceed to integrate the motion of this ray using a $k^{th}$ order integrator using a time step of $\Delta s$. Let us describe what we mean by such an integrator. Suppose we are at the $n^{th}$ step of integration, then the $(n+1)^{th}$ coordinates are given by a function $\vec{I}$ which obeys the properties:

$$\begin{aligned} \vec{u}_{n+1} &= \vec{I}(\vec{u}_n; \Delta s) \\ &= \underbrace{\vec{E}(\vec{u}_n; \Delta s)}_{\text{exact solution}} + \underbrace{\vec{A}(\vec{u}_n; \Delta s)\ \Delta s^{k+1}}_{\text{Leading order error}} + \cdots \text{higher order in } \Delta s \end{aligned} \tag{6}$$

The function $\vec{E}$ in Equation (6) is the exact trajectory of $\vec{u}_n$ at time $s_n = n\Delta s$ for a time $\Delta s$. Suppose we use this method $\vec{I}$ to integrate for $N$ identical steps for a time $s_N = N\Delta s$. Then one can show that the final result $\vec{u}_N$ obeys the following rule:

$$\begin{aligned} \vec{u}_N &= \underbrace{\vec{E}(\vec{u}_0; N\Delta s)}_{\text{exact solution}} + \underbrace{\vec{B}\,\Delta s^k}_{\text{Leading order error}} + \cdots \text{higher order in } \Delta s \\ &= \vec{E}_N + \vec{B}\,\Delta s^k \end{aligned} \tag{7}$$

## 3.2 The optical or Darbousian limit

If we repeatedly apply the integrator for a given $\Delta s$ on the initial ray of Equation (5), two things can happen after a time $N\Delta s$.

1. The particle in the region $0 < z < L$ makes a U-turn and travels in the negative $z$ direction. Time integration allows for this situation. Of course in a standard straight multipole it usually means the end of the beam and tracking should be stopped. The same particle, in a exact code such as PTC or TEAPOT, would produce insane results such as complex longitudinal momentum. In such codes tracking must be stopped for mathematical reasons. In less exact codes, such as SixTrack [4], which use the small angle approximation, thing will look normal but of course the results obtained must be completely worthless.

2. The particle at time $N\Delta s$ reached the vicinity of the boundary of the magnet. If one more step is applied for a time $\Delta s$, then the particle crosses over the boundary. This is the case of interest in optics.

Thus we must investigate how is one suppose to perform a calculation of the approximate intersection of the ray with the boundary. We must now propagate $\vec{\omega}_N$ for an unknown time $\delta s$ such that:

$$\vec{\omega}_{n\Delta s+\delta s} = (x, w_1, y, w_2, L, w_3, s_0 + n\Delta s + \delta s, \delta) \tag{8}$$

There are two criteria on which we will insist for our algorithm.

- We have so far used a $k^{th}$ order algorithm for the computation of the moment. Whatever we do on this last critical step should not deteriorate the integrator. What is the point of using a fancy method of integration if the last step destroys the very sophistication we painfully programmed into our tracking code?

- The first step will involve solving an equation for the final time step $\delta s$. We must insure that this method is polymorphic: it carries correctly the Taylor series computation. This is only important in a code where polymorphism is used such as PTC.

Let us assume that we have a method, which could be an integrator, and which is locally of order $\chi$ to estimate $\delta s$.

We will now compare the exact intercept with the final surface $z = L$ with the one provided by our integrator equipped with an order $\chi$ intersection procedure. Denoting the exact extra time needed at the very end by $\widetilde{\delta s}$, we must have:

$$E_5\left(\vec{E}_N; \widetilde{\delta s}\right) \;=\; L \tag{9}$$

We write the equivalent equation for integrator. Notice that we introduce the error of order $\chi$ of the procedure.

$$E_5\left(\underbrace{\vec{E}_N + \vec{B}\Delta s^k}_{\vec{u}_N}; \delta s\right) + C_5 \delta s^\chi \;=\; L \tag{10}$$

We use Equations (9) and (10) to solve for $\widetilde{\delta s}$ in terms of the computed value $\delta s$:

$$\widetilde{\delta s} \;=\; \delta s + \left(\frac{\partial E_5}{\partial \delta s}\right)^{-1} \left\{\vec{\nabla} E_5 \cdot \vec{B}\, \Delta s^k + C_5 \delta s^\chi\right\} \tag{11}$$

We finally express an arbitrary final coordinate:

$$\begin{aligned}
E_{\mathrm{i}}^{\mathrm{exact}} &= E_i\left(\vec{u}_N; \delta s\right) - \vec{B}\cdot\vec{\nabla} E_i \Delta s^k \\
&\quad + \frac{\partial E_i}{\partial \delta s}\left\{\left(\frac{\partial E_5}{\partial \delta s}\right)^{-1}\left\{\vec{\nabla} E_5 \cdot \vec{B}\, \Delta s^k + C_5 \delta s^\chi\right\}\right\} \\
E_i\left(\vec{u}_N; \delta s\right) - E_{\mathrm{i}}^{\mathrm{exact}} &= \underbrace{\Delta s^k \left\{\vec{B}\cdot\vec{\nabla} E_i - \frac{\partial E_i}{\partial \delta s}\left(\frac{\partial E_5}{\partial \delta s}\right)^{-1} \vec{\nabla} E_5 \cdot \vec{B}\right\}}_{\text{Body term}} \\
&\quad \underbrace{- \frac{\partial E_i}{\partial \delta s}\left\{\left(\frac{\partial E_5}{\partial \delta s}\right)^{-1} C_5\right\} \delta s^\chi}_{\text{Boundary term}}
\end{aligned} \tag{12}$$

Equation (12) is a fundamental result of time integration in Darbousian systems.

In standard accelerator physics, where integration is "Darbousian", the variable $\Delta s$ is an integer fraction of the length $L$ and the integration reaches the boundary perfectly and we need not to worry about boundary effects.

However, in Zgoubi as well as in other codes using true time, the second term proportional to $\delta s^\chi$ can be critical. Let us assume that someone wrongly assumed that the last step is small

and therefore felt justified to choose a sloppy method to find the intercept— for example some quadratic interpolation resulting in a $\chi$ of two. In that case, since the variable $\delta s$ is statistically of order $\Delta s$, the ratio of the tiny boundary term to the body term, goes like $\Delta s^{\chi-k}$ which in standard Zgoubi would be $\Delta s^{-2}$ since the code runs by default with $k = 4$.

This means that for small time steps poorly implemented boundaries **totally** ruin the convergence properties of the integrator chosen. This is a somewhat counter intuitive phenomenon. As the time step is reduced, the integrator will start to behave poorly in terms of speed of convergence.

We have verified that to be the case for the Zboubi methods as well as for the Runge-Kutta methods. Conversely, RK4 which is a fairly robust method behaves very well for analytical models even surpassing the methods in Zboubi. We surmise that its bad reputation in some milieu is related to a lack of understanding of the boundary effects. Generally speaking, it suffices that one step of integration be mishandled to ruin an integration method.

Finally when the field is known numerically and our ability to subdivide in a Maxwellian way is limited if not nil, the comments of this section are absolutely critical. We have found for example that Darbousian integration using a Tosca data files works very well with RK4. In addition, Zboubi style integration with a careful boudary evaluation [5] also works very well.

## 3.3 Simple solution for the boundary problem

When solving for the quantity $\delta s$, we need a model for the dynamics which is locally no worse than one order less than our integrator as shown in Equation (12). One possible solution which works very well for explicit Runge-Kutta and the Zgoubi integration method is to use the very method used in the body! In that case, Equation (10) takes the form

$$I_5\left(\vec{u}_N; \delta s\right) \;\; = \;\; L \tag{13}$$

This equation can be solved by the following algorithm:

$$\begin{aligned}
&\delta s = 0 \; ; \; s = s_N ; \; \vec{u} = \vec{u}_N ; \\
&\text{do until } u_5 = L \\
&\quad \delta s = \delta s + \left(L - u_5\right) / u_6 \\
&\quad \vec{u} = \vec{u}_N ; \; s = s_N \\
&\quad \vec{u}_s = \; \vec{I}\left(\vec{u}_N; s, \delta s\right) \\
&\text{end do}
\end{aligned} \tag{14}$$

The variable $s$ is now an output of the calculation. Conversely, the variable $z = u_5$ is now fixed at the value $z = L$ as it should be in a Darbousian setting. Thus the procedure $\vec{I}$ is of order $\chi = k + 1$ and therefore it is an acceptable procedure for reaching the boundary. This is what Zgoubi does: it uses its own integration method in a procedure called "ITER" to search for the intercept.

We now address the last issue: the production of Taylor maps though polymorphism. If the initial "transverse ray," that is to say, the first four components of $\vec{u}$, are initialized as a Taylor series in the usual manner:

$$u_{0;i} = r_{0;i} + x_i \quad ; \; i = 1, 4 \tag{15}$$

In Equation (15), the vector $\vec{r}_0$ represents the real initial component of the polymorphic vector $\vec{u}_0$. In a code computing only real variables, there are no differences between $\vec{r}_0$ and $\vec{u}_0$. The

variables $x_i$ represent the Taylor map which is carried around the real orbit. For example, if we are computing a map for this single element, it represent the monomials $(x_1, x_2, x_3, x_4)$ and thus Equation (15) is the identity map expressed in coordinates relative to the original orbit. The variables $(u_5, u_6) = (z, w_3)$ in the Darbousian picture are independent variables. In fact, in the last step we make sure that the trajectory ends up at $z = L$! Obviously, the time $s$ becomes the dependent variable and this dependence is computed entirely in the last step. How do we make sure that the Taylor depedence is carried correctly to all variables, i.e., to $(u_1, u_2, u_3, u_4, s)$? The answer is simple: we must make sure that the time variable in all our calculation is a polymorph. This means that in the algorithm (14), we must declare the variable $\delta s$ as well as all the time-like variables in the integrator $\vec{I}$ to be polymorphs. This is not something we do automatically in a Darbousian integrator since the length is fixed for a given magnet.

# 4 Zgoubi using PTC's Structures

In this section, we describe how Zgoubi, and indeed any other code, can use PTC's structure. This work is complete on the PTC side: it is not done properly on the Zgoubi side owing to the complex structure or more precisely lack of structure of Zgoubi. It is a real messy job to extract a well-defined propagator out of Zgoubi. The code is *definitely not* object oriented in spirit.

## 4.1 PTC's Structures: Why?

Most people associate PTC wrongly to Taylor maps. But Taylor maps are in PTC a simple consequence of the usage of a real(8)/Taylor polymorphic Fortran90 type. PTC is above all an integrator. In that sense it does not differ from Zgoubi except that Zgoubi has nearly two hundred types of elements, and therefore it is exponentially more complete than PTC.

So, from a purely physics standpoint, particularily in small machines where one can afford slower calculations, Zgoubi is certainly a very useful tool.

PTC is unique for only one thing: it does not describe a beam line as a collection of magnet. In PTC, the beam line, called LAYOUT, is a link list of containers, called FIBRES. Each fibres has geometric patches to previous fibres as well as misalignments operators. It also has a pointer to the magnet propagator.

Therefore tracking in PTC proceeds fibre to fibre. When it enters a fibre, the ray

- is first transformed geometrically by the patches and misalignments,
- is then transformed by the Darbousian map of the magnet, whatever that may be,
- is finally transformed geometrically to exit at the proper location by more patches and misalignents.

This flexibility and the existence of pointers, permit the sharing of elements through beam lines and above all the exact positioning of elements.

In PTC, one can place the magnet in the ring as the engineers would do during installation. The code can then be instructed to compute the geometric patches. The survey of PTC takes into account the patches and the internal geometry of the magnet represented by its map. If patching is not properly done, the survey displaces the fibres, in other words, the fiducial location of the magnets. Thus the survey command of PTC provides a link between the global word and the Darbousian world.

### 4.2 Zgoubi Inclusion

We can schematically represent the map through a fibre of PTC as follows:

$$F = P_{out} \circ M_{out} \circ T \circ M_{in} \circ P_{in} \tag{16}$$

The map $T$ represents the "ideal magnet." The patches $P_{in,out}$ are dynamical version of the geometric operators connecting the present fibre with its predecessor or successor. In PTC, the magnet that follows the present magnet depends on the fibre. This feature permits the exact construction of recirculators and is precisely the reason for the introduction of the fibre. Other codes which use linked lists, such as MAD-X, create linked lists of magnets which is an unfortunate shortcoming. The operators $R_{in,out}$ represent the misalignments. They are of the same nature as the patches.

The idea of using PTC structures is simple. One simply replaces $T$ by a call to a different program, namely Zgoubi. This sounds easy in theory especially since Zgoubi is in Fortran77. However, as we pointed out, Zgoubi inner programming structure is archaic in the extreme. It does not distinguish magnets from commands or other beam attributes. Internally an array $A(N, M)$ contains all the potential $N$ items of the input files with its potential $M$ attributes. Worse even, there are no set meaning for an attribute: for example $A(N, 2)$ can contain a magnet length for a certain type of magnet while for another type this might be stored in $A(N, 3)$. In fact the situation is even worse than that: depending on the input chosen for a given magnet, the location of crucial parameters changes location. Younger people who have been trained in the object oriented method must wonder how such a thing can ever occur or be allowed to persist. It must remembered that codes like Zgoubi originated in the 1970s and that furthermore it is a rather complete code model wise particularly when compared to existing programs and therefore a rewriting of Zgoubi would come at a steep price.

We tested a prototype of Zgoubi using PTC structures. A real inclusion would require more work.

## 5 Appendix: Comparisons of Tracking Plots and the subroutine ITER

The test lattice is the FD cell of the EMMA non-scaling, linear lattice electron model. Sharp edge magnets are considered here.

Goal : show that all usual integration methods, symplectic or not, yield comparable behavior up to stability limits (RK4-z or -time with ITER, RK6-z, Zgoubi4 or 6-time with ITER, Ruth4-z, Yoshida6-z).

The step size is chosen as large as possible, such that (i) no noticeable spread of the invariant is observed, and that (i) doubling the step size would cause significant spread/thickness.

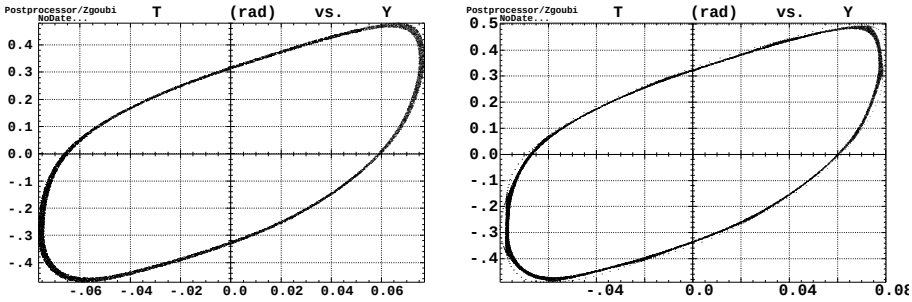


Figure 1: Zgoubi, order 4 with 20 steps (left) or 6 with 5 steps (right).

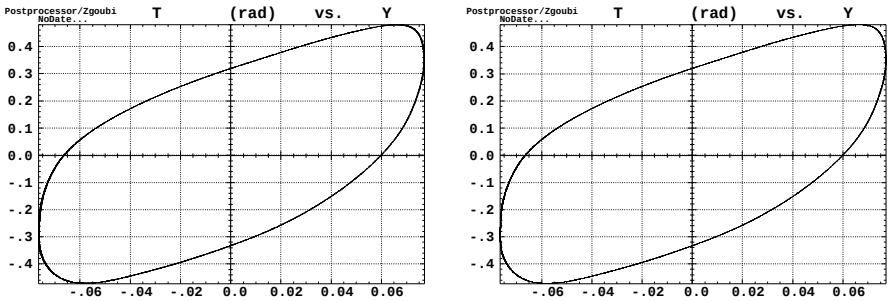


Figure 2: Zgoubi, order 4 with 40 steps (left) or 6 with 10 steps (right).

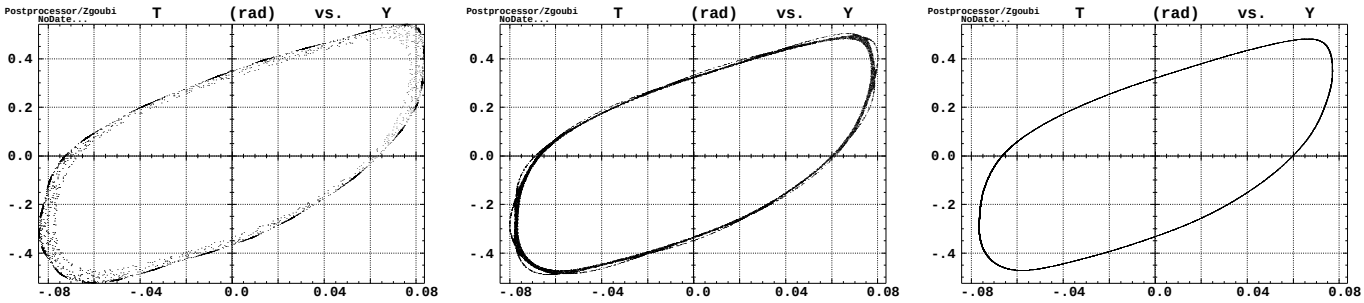


Figure 3: Effect of loss of precision in ITER. Zgoubi is used at order 6 with 20 steps, so to insure good precision at all steps but the last one. (This should produce an invariant at least as good as in Fig. 2-right). Here however we purposely mishandle the last step, lowering the order of ITER ($\chi$) to 2, 3 or 4 from left to right. Conclusion: spoiling the precision in the last step is enough to spoil the overall symplecticity. It decreases the precision to respectively order 2, 3 and 4.

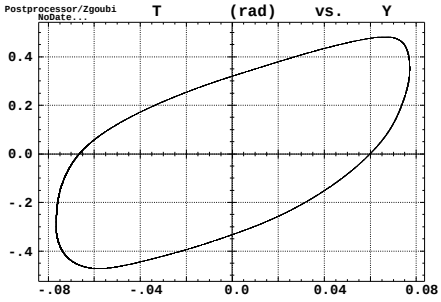


Figure 4: Starts from the conditions in Fig. 3, namely, order 6 at all steps except for the last step where order 2 is taken. The difference is in the number of steps, 2000 instead of 20. The precision is regained. The overall order is unchanged, still spoiled down to order 2 due to the order 2 in the last step. However the accuracy is better in account of the high number of steps.

RK

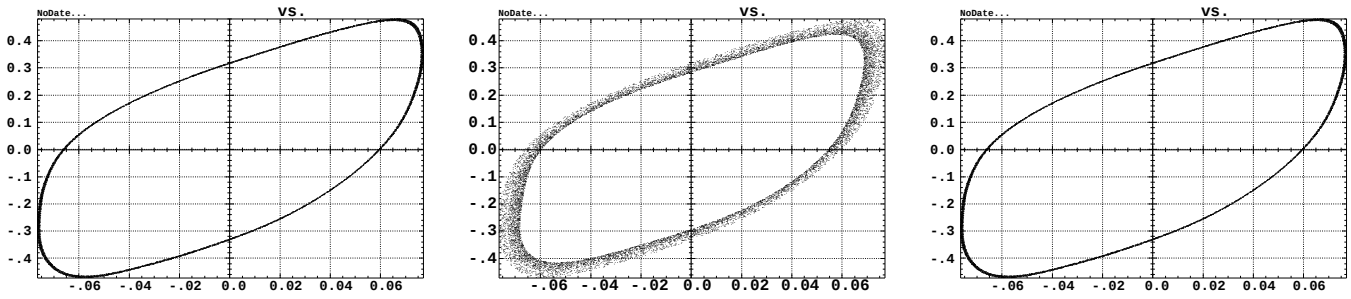


Figure 5: RK4 with 5 steps (left) or RK6 with 5 steps (middle), both with z-integration, i.e., PTC Darboussian integration. RK4 with 5 steps and $s$-integration (right) works better.

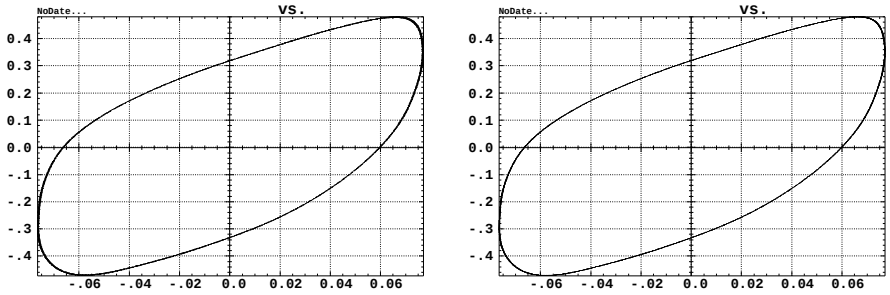


Figure 6: RK4 with 10 steps (left) or RK6 with 10 steps (right). Both z-integration.

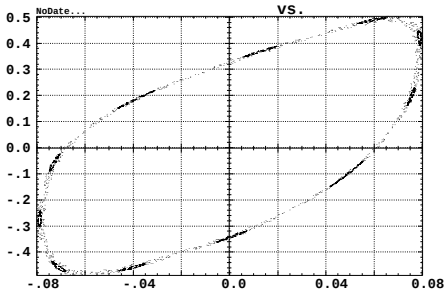


Figure 7: Effect of loss of precision in ITER. RK4 with 10 steps, so to insure good precision at all steps but the last step in ITER uses a $\chi$ of one. Conclusion: spoiling the precision in the last step is enough to spoil the overall symplecticity, it decreases the precision of the integrator to an RK1 integrator.

# References


[1] F. Méot. The ray-tracing code Zgoubi. *NIM*, A 427:353–356, 1999. This code is a plain integrator for rings without special enforcement of symplecticity.

[2] E. Forest, F. Schmidt, and E. McIntosh. Introduction to the Polymorphic Tracking Code. Technical Report CERN-SL-2002-044, KEK-Report 2002-3, 2002.

[3] L. Schachinger and R. Talman. *Part. Accel.*, 22:35, 1987. E. Forest checked TEAPOT against the PSR lattice paper of Dragt for the appendix of this paper.

[4] F. Schmidt. SIXTRACK, version 1.2, single particle tracking code treating transverse motion with synchrotron oscillations in a symplectic manner. Technical Report CERN SL/94–56 (AP), CERN, 1994.

[5] M. Aiba and F. Méot. Determination of KEK 150 MeV FFAG parameters from ray-tracing in TOSCA field maps. Technical Report CEA DAPNIA-04-188 and CERN-NUFACT-Note-140, CEA/CERN, May 2006.