Tracking Codes

oPAC Advanced School on Accelerator Optimization

Dr David Newton

University of Liverpool and The Cockcroft Institute

9th July 2014

Tracking Code Overview

There are a huge number of tracking codes available for different applications

- Optics
- Particle Tracking
- Space Charge
- Linacs
- EM solvers
- Synchrotron Radiation
- Specific processes

How do we determine the correct code for a specific problem?

Tracking Code Overview

- The choice of which code to use depends on the required accuracy, computation speed, reputation, ease of use, what your collaborators use ... etc
- it's very specific to the problem you're trying to solve
- I won't be talking in any great detail about specific codes
- I'll concentrate on the underlying algorithms that many of the codes use and the approximations that are made
- I'll try to explain how these methods were developed which gives some insight into the underlying assumptions made

A simple definition of particle tracking could be:

Given a particle's phase space coordinates at the beginning of an accelerator element, calculate the particle's trajectory through the element

The Problem

- The Laws of Motion for a Newtonian dynamical system reduce to a set of second-order ordinary differential equations
- i.e. The Lorentz force equation:

$$\ddot{\mathbf{x}} = \frac{q}{m}(\dot{\mathbf{x}} \times \mathbf{B} + \mathbf{E})$$

- or more generally:

$$\ddot{\mathbf{q}} = f(\mathbf{q}, \dot{\mathbf{q}}; t)$$

- where **q** refers to the instantaneous position of a particle, and the dot represents the derivative w.r.t time, t.

Do these equations actually contain information about trajectories?

The Problem

- Note that we can convert any second-order differential equation into two first-order differential equations :

$$\ddot{q} = f(q, \dot{q}; t) \tag{1}$$

by the rules,

$$y_1(t) = q(t),\ y_2 = \dot{q}(t)$$

Equation (1) is then equivalent to the first-order set

$$\dot{y}_1 = y_2 \tag{2}$$
$$\dot{y}_2 = f(q, \dot{q}; t) = f(y_1, y_2; t) \tag{3}$$

The General Problem

- The solution to the Cauchy (or initial value problem) states: For any set of first-order differential equation of the form

  $$\dot{y_j} = f_j(y_1, y_2, \ldots, y_m; t), \;\; j = 1, \ldots, m$$

  if f_j and the partial derivatives $\partial f_j / \partial y_k$ exist and are continuous in some region R and for some time t in an interval T about a fixed value t^0, there exists a unique solution

  $$y_j(t) = g_j(y_1^0, \ldots, y_m^0; t^0; t)$$

  with the property

  $$y_j(t^0) = g_j(y_1^0, \ldots, y_m^0; t^0; t^0) = y_j^0$$

Do these equations actually contain information about trajectories?...

YES! They contain all the information about the trajectory

The Problem

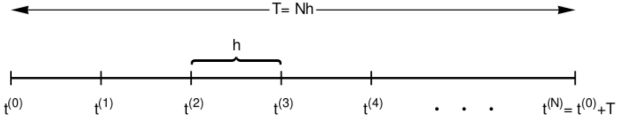
- The solution is guaranteed to exist for a finite interval of time about the point t^0 and can be extended forward or backward in time as long as the f_j are continuous and the $y_j(t)$ remain within a region R.
- The first-order differential equations constitute a set of ‘marching orders’
- Once the initial time t^0 and the initial starting point y^0 are specified, the trajectory is completely determined
- The trajectory can be determined by solving the set of first-order differential equations

The Problem

- Unfortunately, there are very few systems that have closed form analytical solutions. However, a complete knowledge of all possible allowed trajectories is not necessary.
- It often suffices to have a qualitative description of the types of allowed motion supplemented by a detailed knowledge of representative ‘orbits’ - most easily obtained by numerical integration.

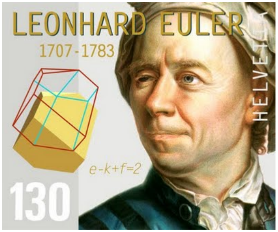
Numerical Integration

- If we write our first-order equations as $\mathbf{y} = (y_1 = q, y_2 = \dot{q})$ and $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, t)$
- We'd like to integrate $\dot{\mathbf{y}}$ from some initial time t_0 to some time T
- We start by dividing T into N equals steps of duration h, so that $Nh = T$ and $t_n = t_0 + nh$



- h should be small compared to the characteristic time scale of the physical system we are studying (i.e. for a pendulum problem, h should be much smaller than the oscillation period)
- Our goal is to find $\mathbf{y}_n = \mathbf{y}(t_n)$

Euler Integration



- Leonhard Euler (1707-1783): Swiss mathematician and Physicist
- The Euler method (1768-1770) is a basic method if integrating differential equations, given an initial value.

Euler Integration

- We know that the solution vectors $\mathbf{y}_n$ exist and are uniquely specified by $\mathbf{y}_0$
- Given $\mathbf{y}_0$ we proceed one step at a time using Taylor expansions:

$$\mathbf{y}_1 = \mathbf{y}(t_1) = \mathbf{y}(t_0 + h) = \mathbf{y}_0 + h\dot{\mathbf{y}}_0 + \mathcal{O}(h^2)$$

or

$$\mathbf{y}_1 = \mathbf{y}_0 + h\mathbf{f}_0 + \mathcal{O}(h^2)$$

- We know $\mathbf{y}_0$ and t_0 so we can compute $\mathbf{f}_0$ and $\mathbf{y}_1$ (ignoring the $\mathcal{O}(h^2)$ error)
- Further steps can be calculated using the rule:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}_n$$

A Numerical Example

Consider the differential equation

$$\ddot{x} + x = 2t$$

with initial conditions $x(0) = 0$, $\dot{x}(0) = 1$
We can convert this into a first-order set by writing $y_1 = x,\ y_2 = \dot{x}$, and find

$$\begin{aligned} f_1(\mathbf{y}, t) &= \dot{y}_1 = y_2 \\ f_2(\mathbf{y}, t) &= \dot{y}_2 = 2t - y1 \end{aligned}$$

and

$$\begin{aligned} \mathbf{y}_0 &= (0, 1) \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h\mathbf{f}_n \end{aligned}$$

A Numerical Example

Explicitly, given the initial coordinates $\mathbf{y}_0 = (y_{1,0}, y_{2,0})$ we calculate for the first step

$$\begin{aligned} f_1 &= y_{2,0} \\ f_2 &= 2t_0 - y_{1,0} \end{aligned}$$

to find the values at the second step:

$$\begin{aligned} y_{1,1} &= hf_1 \\ y_{2,1} &= hf_2 \end{aligned}$$

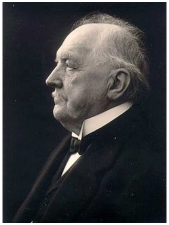
we then update the time $t_1 = t_0 + h$, $\mathbf{f} = (f_1, f_2)$ using the updated values $y_{1,1}, y_{2,1}$ and t_1 and iterate n times

A Numerical Example

- If we step the time to $t_0 + T$ in N steps, at each step we make a local error of order h^2
  - Note that the error is of order h^2 - it is proportional to h^2 with an unspecified proportionality constant
- The cumulative error is of order Nh^2
  - The Euler Method is a first order integrator
- We see that if the step size h is made sufficiently small the error in $\mathbf{y}(t_0 + T)$ can be made arbitrarily small (in principle)
- The Euler method is a rather crude integrator that evaluates $\mathbf{f}$ once at each step - it could be improved by increasing the number of evaluations at each step

Runge-Kutta Integration

- Runga-Kutta methods are an important family of iterative methods used to numerically solve ordinary differential equations
- They were developed around 1900 by the German mathematicians M. Kutta and C. Runge


Martin Wilhelm Kutta
(1867 - 1944)

Carl David Tolmé Runge
(1856 - 1944)

Runge-Kutta Integration

- The general idea of Runge-Kutta methods is to evaluate $\mathbf{f}$ at several points in each step and take a weighted average of the results in such a way that $\mathbf{y}_{n+1}$ is correctly estimated up to some error that is proportional to some power of h (and therefore quite small)
- Deciding which points to use in $\mathbf{f}$ and how to weight them is a complicated procedure (see the references at the end if you're interested)
- There a huge number of different Runge-Kutta methods
- The methods are labelled by the order of the local accuracy - an mth order method is:
    - locally correct through order h^m
    - has local errors of order h^{m+1}

Runge-Kutta Integration

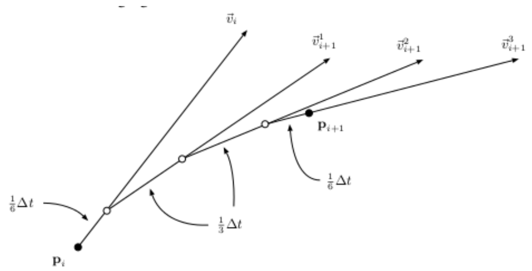
- One method is the RK3 method
  - locally correct through order h^3
  - local errors of order h^4

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{1}{6}(\mathbf{a} + 4\mathbf{b} + \mathbf{c})$$

where

$$\begin{aligned}
\mathbf{a} &= h\mathbf{f}(\mathbf{y}_n, t_n) \\
\mathbf{b} &= h\mathbf{f}(\mathbf{y}_n + \frac{\mathbf{a}}{2}, t_n + \frac{h}{2}) \\
\mathbf{c} &= h\mathbf{f}(\mathbf{y}_n + 2\mathbf{b} - \mathbf{a}, t_n + h)
\end{aligned}$$

- Higher order methods are available
  - The higher the order the more work must be done in the computation



- Several trajectories are predicted and each prediction is used to correct the initial estimation
- An example of a predictor-corrector method

Runge-Kutta Integration

- We mentioned earlier that in principle the error can be made arbitrarily small by decreasing the step size h
- Practically, we're limited by the finite precision of the computation

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}_n$$

- If the h becomes too small, the quantity $h\mathbf{f}_n$ becomes much smaller than $\mathbf{y}_n$ and the local error in $\mathbf{y}_{n+1}$ is dominated by 'round-off' errors.
    - If the sign of each error is random the cumulative error grows as $\sqrt{N}$
    - If the sign is systematic the cumulative error grows as N

A Numerical Example

Consider the differential equation we saw earlier:

$$\ddot{x} + x = 2t$$

with initial conditions

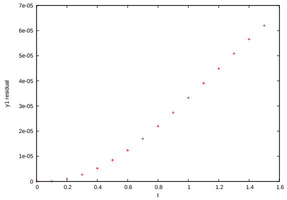
$$x(0) = 0,\ \dot{x}(0) = 1$$

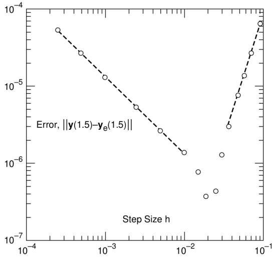
In this case the differential equation is sufficiently simple that we can integrate it analytically to give the exact result:

$$\begin{aligned} y_1 &= x(t) = 2t - \sin(t) \\ y_2 &= \dot{x}(t) = 2 - \cos(t) \end{aligned}$$

Choosing a step size of $h = 0.1$ and integrating from $t = 0$ to $t = 1.5$, the RK3 method can be used to compare the accuracy of the results

A Numerical Example (RK3)





- The plot of the residuals (top) shows an accuracy of $\sim 10^{-4}$ is achieved using a rather modest number of steps ($h = 0.1$)
- The accuracy as a function of step size h is shown (lower). The error decreases (with a gradient of ~ 3) as the step size decreases. As the step-size becomes too small, the numerical round-off errors dominate the calculation and the error grows linearly with N

Numerical Integration

Note that if we can use Hamiltonian mechanics (rather than Newtonian):

$$H = c\sqrt{(\mathbf{p} - q\mathbf{A})^2 + m^2c^2} + q\phi$$

the equations of motion are given directly as two first-order differential equations

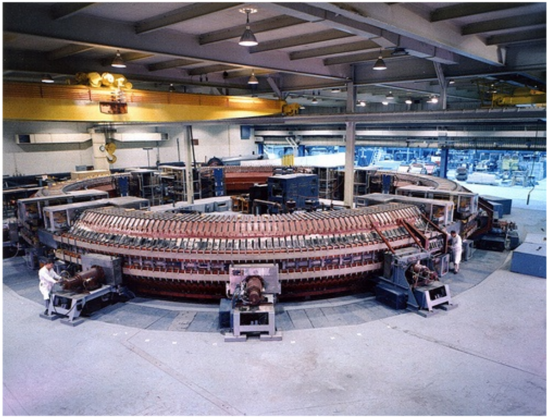
$$\frac{dx_i}{dt} = \frac{\partial H}{\partial p_i}, \ \frac{dp_i}{dt} = -\frac{\partial H}{\partial x_i}$$

which can be integrated numerically using the same procedures

Numerical Integration

- Numerical integration provides a robust method of tracking through arbitrary electro-magnetic fields, but...
- It is computationally expensive
- there is a trade off between the accuracy/time/round-off which you should be aware of
- Certain sets of first-order equations are ‘stiff’ - numerically unstable to integration unless the step-size is very small (even if the underlying solution is smooth)
- It would be obviously beneficial if the equations of motion could be integrated analytically to give a ‘more-or-less exact’ solution...

Strong Focusing



The Cosmotron (1952): Brookhaven's 3 GeV weak-focusing proton synchrotron

Strong Focusing

Courant, Livingstone and Snyder (l-r)

"By reversing the direction of some of the Cosmotrons C-shaped magnets (originally arranged facing outward around the machines circular track), a young Brookhaven physicist named Ernest Courant calculated that the resulting proton beam would focus much more tightly. The strong-focusing principle was born." [1]

[1] Taken from http://scienceblogs.com/brookhaven/2010/06/25/finding-focus-for-the-worlds-a/

Strong Focusing

“ Livingston has said that the idea of strong focusing arose [when], he asked Courant to consider whether they could turn some bending magnets of the new Cosmotron around... Operation at the highest energies was limited by changes in the relative gradient n. Livingston suggested turning some magnets around to cancel this variation of gradient and asked whether this could be done without terrible damage to the focusing of the beam. Courant ... found that focusing was in fact improved and that alternating the focusing could lead to an entirely new class of accelerators. ”

F.T.Cole - “O Camelot! A Memoir of the MURA Years”, 1994

The Equations of Motion for a drift space

Lets start with the simplest example: a field free drift region, of length L. The Hamiltonian (in accelerator coordinates) is:

$$H = \frac{\delta}{\beta_0} - \sqrt{\left(\delta - \frac{1}{\beta_0}\right)^2 - p_x^2 - p_y^2 - \frac{1}{\beta_0^2\gamma_0^2}} = \frac{\delta}{\beta_0} - d$$

with the transverse equations of motion:

$$\begin{aligned} \frac{dp_x}{ds} &= -\frac{\partial H}{\partial x} = 0 \\ \frac{dx}{ds} &= -\frac{\partial H}{\partial p_x} = \frac{p_x}{d} \end{aligned}$$

- Since the momenta, the energy (and d) are all constant these can be integrated directly

The Equations of Motion for a drift space

The exact solutions for the equations of motion in a drift space are:

$$x_1 = x_0 + \frac{p_{x0}}{d} L$$

$$p_{x1} = p_{x0}$$

$$\text{where}, d = \sqrt{\left(\delta - \frac{1}{\beta_0}\right)^2 - p_x^2 - p_y^2 - \frac{1}{\beta_0^2 \gamma_0^2}}$$

- Note that the equations are non-linear - the final values for the co-ordinates have a non-linear dependence on the momenta

The Equations of Motion for a drift space

- A drift space is one of the (very) few examples of a Hamiltonian that can be solved exactly
- In nearly all other cases of accelerator components we have to make some approximations to obtain a convenient form for the solutions for the equations of motion
- The solution can be linearised by expanding the solutions as power series to first order in the dynamical variables...
- Alternatively, the Hamiltonian can be expanded to second order in the dynamical variables (the paraxial approximation).
- Its generally a better approach to find an exact solution to an approximate Hamiltonian, rather than an approximate solution to an exact Hamiltonian
    - If this is done carefully, symplecticity can be preserved

Matrix Tracking

Assuming the fields can be described by a vector potential with no transverse dependence ($\mathbf{A} = (0, 0, A_s)$) where the scalar potential goes to zero ($\phi = 0$), A general Hamiltonian can be written as

$$H = \frac{\delta}{\beta_0} - (1 + hx)\sqrt{\left(\delta + \frac{1}{\beta_0}\right)^2 - p_x^2 - p_y^2 + \frac{1}{\beta_0^2\gamma_0^2}} - (1 + hx)a_s$$

$$H \sim \frac{\delta}{\beta_0} + (1 + hx)\left(\frac{p_x^2}{2D} + \frac{p_y^2}{2D} - D - a_s\right) + \mathcal{O}(4)$$

where h is the curvature, $a_s = (q/P_0)A_s$ and $D = \sqrt{1 + \frac{2\delta}{\beta_0} + \delta^2}$. This approximation is valid for

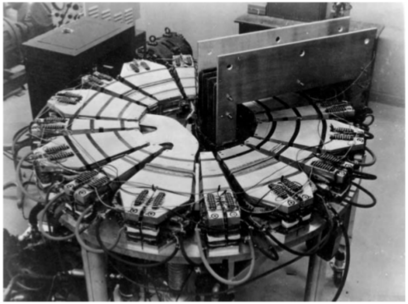
$$\frac{p_x^2 + p_y^2}{2D^2} \ll 1$$

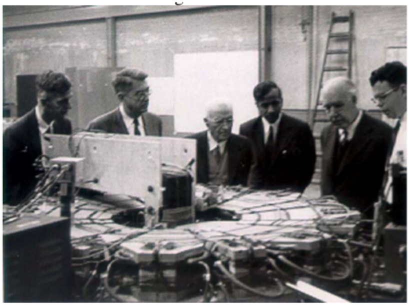
Matrix Tracking

$$H \sim \frac{\delta}{\beta_0} - (1 + hx)\left(\frac{p_x^2}{2D} + \frac{p_y^2}{2D} - D - as\right)$$

- Truncating the Hamiltonian at second order will give linear transfer matrices, $\mathbf{z}_f = R\mathbf{z}_i$
- Truncating at higher order gives the non-linear dynamics
  - i.e. TRANSPORT method used in MAD8

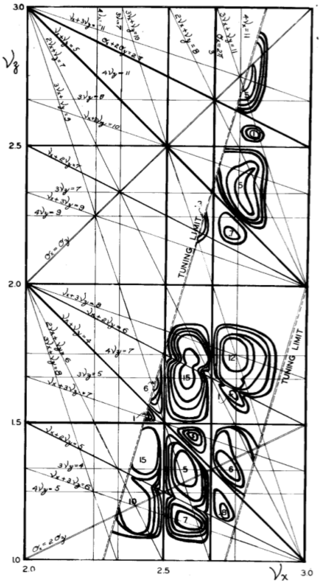
$$z_{j,f} = \sum_k R_{jk} z_{j,i} + \sum_{kl} T_{jkl} z_{j,i} z_{l,i} + \dots$$





- At around the same time that strong-focusing was discovered, a prototype FFAG was being studied at MURA
- Jackson Laslett was using an early IBM computer with a Runge-Kutta integrator to study the non-linear particle dynamics in this machine.
- 'the chief advantage of our theory was that it was sufficiently complicated that it was hard to show it will not work'*

K.R.Symon, 'MURA Days', PAC 2003, WOPA003. In fact the model worked very well



- With his code Laslett was able to construct a Frequency Map of the FFAG dynamics
- Studied early examples of chaotic motion
- He was aware that the RK method didn't conserve phase-space: 'non-Liouvillean errors were present'
- Were these chaotic motions 'real' or an artefact of the integration method?

Symplectic Integration Schemes

Suppose we have a function of $2N$ phase space variables

$$f = f(x_i, p_i)$$

such that f has no explicit dependence on the independent variable, s. However the value of f will evolve with s, because the values of the dynamical variables evolve with s:

$$\frac{df}{ds} = \sum_{i=1}^{N} \frac{dx_i}{ds}\frac{\partial f}{\partial x_i} + \frac{dp_i}{ds}\frac{\partial f}{\partial p_i}$$

Using Hamilton's equations, this becomes:

$$\frac{df}{ds} = \sum_{i=1}^{N} \frac{\partial H}{\partial p_i}\frac{\partial f}{\partial x_i} - \frac{\partial H}{\partial x_i}\frac{\partial f}{\partial p_i}$$

Symplectic Integration Schemes

$$\frac{df}{ds} = \sum_{i=1}^{N} \frac{\partial H}{\partial p_i}\frac{\partial f}{\partial x_i} - \frac{\partial H}{\partial x_i}\frac{\partial f}{\partial p_i}$$

If we define an operator $:g:$ for any function $g(x_i, p_i)$:

$$:g: = \sum_{i=1}^{N} \frac{\partial g}{\partial x_i}\frac{\partial}{\partial p_i} - \frac{\partial g}{\partial p_i}\frac{\partial}{\partial x_i}$$

then, as long as f and H have no explicit dependence on s, we can write the evolution of f as

$$\frac{df}{ds} = -:H:f$$

Symplectic Integration Schemes

$$\frac{df}{ds} = \sum_{i=1}^{N} \frac{\partial H}{\partial p_i}\frac{\partial f}{\partial x_i} - \frac{\partial H}{\partial x_i}\frac{\partial f}{\partial p_i}$$

If we define an operator $: g :$ for any function $g(x_i, p_i)$:

$$: g := \sum_{i=1}^{N} \frac{\partial g}{\partial x_i}\frac{\partial}{\partial p_i} - \frac{\partial g}{\partial p_i}\frac{\partial}{\partial x_i}$$

then, as long as f and H have no explicit dependence on s, we can write the evolution of f as

$$\frac{df}{ds} = - : H : f$$

$: H :$ is a Lie operator

Lie Transforms

Now suppose we expand, the function f about the point s_0:

$$\begin{aligned} f|_{s_0+\Delta s} &= f|_{s_0} + \Delta s \left.\frac{df}{ds}\right|_{s_0} + \frac{\Delta s^2}{2} \left.\frac{d^2 f}{ds^2}\right|_{s_0} + \dots \\ &= \sum_{n=0}^{\infty} \frac{\Delta s^n}{n!} \left.\frac{d^n f}{ds^n}\right|_{s_0} \end{aligned}$$

Lie Transforms

Now suppose we expand, the function f about the point s_0:

$$\begin{aligned} f|_{s_0+\Delta s} &= f|_{s_0} + \Delta s \left.\frac{df}{ds}\right|_{s_0} + \frac{\Delta s^2}{2} \left.\frac{d^2 f}{ds^2}\right|_{s_0} + \dots \\ &= \sum_{n=0}^{\infty} \frac{\Delta s^n}{n!} \left.\frac{d^n f}{ds^n}\right|_{s_0} \end{aligned}$$

which we can write as,

$$f|_{s_0+\Delta s} = e^{\Delta s \frac{d}{ds}} f|_{s_0}$$

Lie Transforms

Now suppose we expand, the function f about the point s_0:

$$\begin{aligned} f|_{s_0+\Delta s} &= f|_{s_0} + \Delta s \, \frac{df}{ds}\bigg|_{s_0} + \frac{\Delta s^2}{2} \, \frac{d^2 f}{ds^2}\bigg|_{s_0} + \ldots \\ &= \sum_{n=0}^{\infty} \frac{\Delta s^n}{n!} \frac{d^n f}{ds^n}\bigg|_{s_0} \end{aligned}$$

which we can write as,

$$f|_{s_0+\Delta s} = e^{\Delta s \frac{d}{ds}} f|_{s_0}$$

or

$$f|_{s_0+\Delta s} = e^{-\Delta s :H:} f|_{s_0}$$

Lie Transforms

- The operator $e^{-\Delta s :g:}$ is known as a Lie Transform, with generator g.
- Applying a Lie Transform with the Hamiltonian as a generator, to a function f produces a transfer map
- The function f can be any function of the dynamical variables
  - If we set it equal to each of the dynamical variables in turn we can construct a full map for the system

Lie Transforms

- This method requires only differentiation - it can be applied even when the equations of motion cannot be analytically integrated
- Any Lie transform represents the evolution of a Hamiltonian system, so it's guaranteed to be symplectic
- The full transfer map is given in the form of a power series, which usually does not have a closed form
  - If the power series is truncated, symplecticity is lost
  - This isn't always important, as long as the power series converges
- The technique can be modified to produce a symplectic map in closed form
  - This usually requires an approximation of the Hamiltonian
  - $\rightarrow$ Symplectic Integrators

Symplectic Kicks

The Hamiltonian for a sextupole magnet, in one degree of freedom, can be written as

$$H = -\sqrt{1 - p_x^2} + \frac{1}{6} k_2 x^3$$

Applying the Lie Transform $e^{-L{:}H{:}}$ (representing a sextupole with length L) produces a power series with infinitely many terms. But we can write the Hamiltonian as the sum of a drift term and a kick term:

$$H = H_d + H_k$$

The Lie transforms $e^{-L{:}H_d{:}}$ and $e^{-L{:}H_k{:}}$ can be written exactly in closed form, so could we approximate the sextupole as a drift followed by a kick and transform each term separately?
i.e.:

$$e^{-L{:}H{:}} = e^{-L{:}H_d + H_k{:}} \simeq e^{-L{:}H_d{:}} e^{-L{:}H_k{:}}$$

Symplectic Kicks

It turns out[2] that

$$e^{-L:H:} = e^{-L:H_d+H_k:} = e^{-L:H_d:}e^{-L:H_k:} + \mathcal{O}(L^2)$$

but because each transform has a closed form solution, neglecting the additional $\mathcal{O}(L^2)$ terms still gives a symplectic representation. A better approximation can be made by writing the Hamiltonian as the sum of three terms (drift-kick-drift):

$$H = \frac{1}{2}H_d + H_k + \frac{1}{2}H_d$$

$$e^{-L:H:} = e^{-L:H_d/2+H_k+H_d/2:} \simeq e^{-L:H_d/2:}e^{-L:H_k:}e^{-L:H_d/2:}$$

which gives a closed form symplectic map , with errors of order L^3 - a second-order symplectic integrator

[2] using the Baker-Campbell-Hausdorff and the Zassenhaus formulae

Symplectic Tracking Timeline

- PATRICIA: 1976 kick code
- RaceTrack: 1984 - Transverse Variable kick code
- SIXTRACK: 1984 - 6D kick code
  - Still used for DA calculations at CERN
- MARYLIE: 1985 - 3rd order Lie Maps (symplectic numerical tracking)
- TEAPOT: 1987 - first kick integrator to use the full Hamiltonian (no paraxial approximation)
- COSYINFINITY:2000 - High order Hamiltonian expansion, symplectification using generating functions
- PTC: 2008 - Full Hamiltonian, symplectic tracking, implemented in MADX, BMAD...

Symplectic Tracking

- Analytical integration using Lie Algebra results in an analytical transfer map
- If the Lie transform has a closed form (kick codes) the resulting map is symplectic
- If the map must be truncated at some high order, symplecticity is lost
  - But the symplectic error can be made very small
  - It gives a very accurate description of the dynamics
- Symplectic integrators are available for s-dependent fields (undulators, fringe fields) - i.e. Wu-Forest Robin
- Numerical integration with these integrators gives a symplectic description of the dynamics (up to machine precision)

Numerical Integration Overview

- Advantages:
  - Robust
  - Can use full Hamiltonian (no approximations)
  - Flexible - arbitrary fields (fringe fields etc.)
  - Symplectic Integrators are available (eg Wu-Forest-Robin) but approximate the Hamiltonian
- Disadvantages:
  - slow
  - Generally not symplectic

Matrix Tracking Overview

- Advantages:
  - Robust - well benchmarked
  - Very fast
  - Linear tracking is symplectic
- Disadvantages:
  - Generally approximates the Hamiltonian (paraxial)
  - Non-linear terms aren't symplectic

Kick-Code Tracking Overview

- Advantages:
  - Robust - well benchmarked
  - Very fast
  - Symplectic
- Disadvantages:
  - Assumes no longitudinal field gradient
  - Approximates the Hamiltonian (how good is the approximation?)

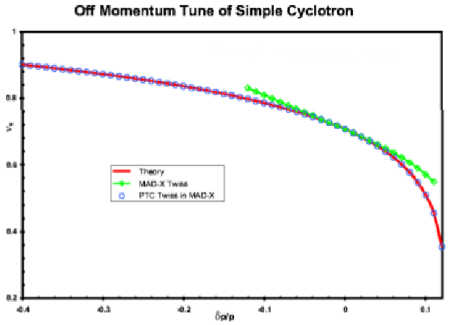
Tracking Methods Overview

- We've covered a lot of different tracking methods so far
- They all have advantages and disadvantages
- Most tracking codes available now, were written for a very specific application, so if you use them
    - Be aware of the approximations used in the code, and make sure they are suitable for your application

Tracking Methods Overview

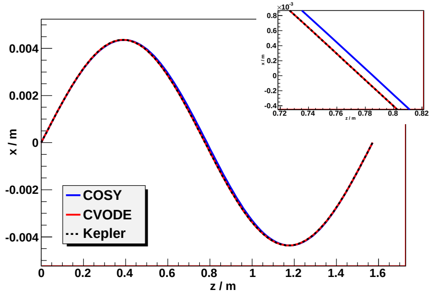
- Relativistic approximation - assumption that $v \sim c$
- small angle approximation - is this valid?
- Energy - does the particle energy change? Is this handled correctly?
- Is symplecticity important?
  - Not only useful for long term tracking
  - If you're studying non-symplectic effects (space-charge, wakefields) it may to be useful to use a symplectic code
- Reference problems - how are the canonical variables defined?
  - Elegant uses x' rather than P_x/P_0
  - Longitudinal variables are often defined differently in different codes

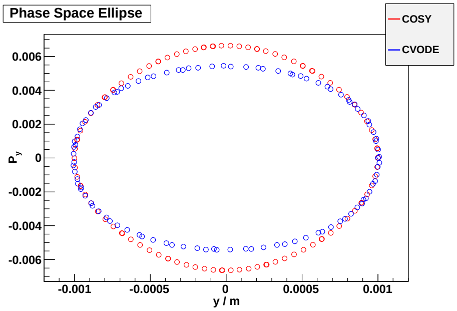
Example: Off momentum tune in cyclotrons



- MAD curve uses the expanded Hamiltonian
- PTC uses the full Hamiltonian
  - (PTC is now implemented in MADX)

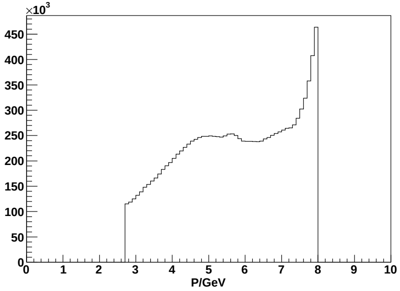
Example: USR @ FLAIR (Antiproton Ring)





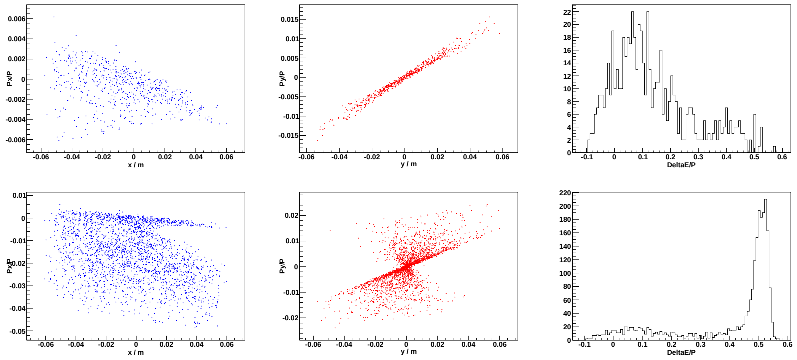
- The USR is a proposal for an electro-static low energy anti-proton ring at FLAIR
- $E \sim 20$ keV $\rightarrow$ non-relativistic, energy changes appreciably in the ES elements
- Comparison with COSY and numerical integrator (CVODE) tracking through a spherical deflector

Example: G-2 Muon decay line



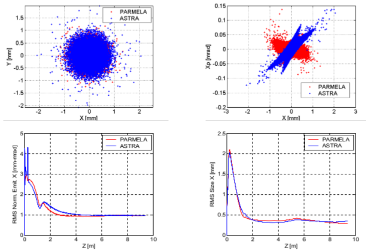
- Immediately after the proton target, the beam is mixture of protons, neutrons, pions etc. with a HUGE momentum spread
- Initially the decay line was implemented as a MAD8 Lattice
- The decay line select particles of the correct rigidity, so its not obvious, looking at the end of the line that there is a problem!

Example: G-2 Muon decay line



- Comparison of tracking up to the first dipole using MAD8 and PTC
- The phase space distributions at the end of the line are similar
- MAD8 predicts $\sim 30\%$ more muons at the end of the decay line
- Also benchmarked with RK integrator and BMAD

Example: ASTRA v PARMELA in the LCLS injector



- Twiss parameters, beam spot-size are very similar
- Very different phase-space distributions
- After a LOT of benchmarking, it turned out that the the longitudinal distributions are defined differently in these codes and weren't being converted correctly
    - In the simulations, the bunches were traversing a linac at different phases, which lead to different space-charge blow-up

Tracking Codes

There have been many tracking codes written over the years...
AT, BETA, BMAD, COMFORT, COSYINFINITY, DIMAD, ELEGANT, LEGO LIAR, LUCRETIA, MAD8,MADX, MARYLIE, MERLIN, ORBIT, PETROS, PLACET, PTC, RACETRACK, SAD, SIXTRACK, SYNCH, TEAPOT, TRACY, TRANSPORT, TURTLE, UAL, ZGOUBI, ...

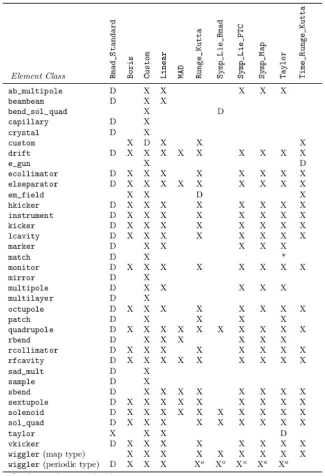
Depending on your application, they may or may not be suitable...

Single Particle Tracking

- MAD8 - TRANSPORT equations, based on expanded Hamiltonian (paraxial approximation)
- ZGOUBI - Numerically integrates the Lorentz equations, truncated Taylor maps
- GPT - 5th order Runge-Kutta
- SIXTRACK - kick code (symplectic, paraxial approximation)
- PTC - Symplectically integrates full Hamiltonian for hard-edge elements (implemented in MADX, BMAD)
- COSY,MARYLIE, BMAD - Truncated Taylor Maps using Differential Algebra (may be 'symplectified')

Tracking Methods in BMAD

- In the BMAD code the tracking method used can be set for individual elements



Element Class	Bmad_Standard	Boris	Custom	Linear	MAD	Runge_Kutta	Symp_Lie_Bmad	Symp_Lie_PTC	Symp_Map	Taylor	Time_Runge_Kutta
ab_multipole	D		X	X				X	X	X	
beambeam	D		X	X							
bend_sol_quad			X				D				
capillary	D		X								
crystal	D		X								
custom		X	D	X		X					X
drift	D	X	X	X	X	X		X	X	X	X
e_gun			X								D
ecollimator	D	X	X	X		X		X	X	X	X
elseparator	D	X	X	X	X	X		X	X	X	X
em_field		X	X			D					X
hkicker	D	X	X	X		X		X	X	X	X
instrument	D	X	X	X		X		X	X	X	X
kicker	D	X	X	X		X		X	X	X	X
lcavity	D	X	X	X		X		X	X	X	X
marker	D		X	X				X	X	X	
match	D		X							*	
monitor	D	X	X	X		X		X	X	X	X
mirror	D		X								
multipole	D		X	X				X	X	X	
multilayer	D		X								
octupole	D	X	X	X		X		X	X	X	X
patch	D		X			X		X		X	
quadrupole	D	X	X	X	X	X	X	X	X	X	X
rbend	D		X	X	X			X	X	X	
rcollimator	D	X	X	X		X		X	X	X	X
rfcavity	D	X	X	X	X	X		X	X	X	X
sad_mult	D		X								
sample	D		X								
sbend	D		X	X	X	X		X	X	X	X
sextupole	D	X	X	X	X	X		X	X	X	X
solenoid	D	X	X	X	X	X	X	X	X	X	X
sol_quad	D	X	X	X		X	X	X	X	X	X
taylor	X		X	X						D	
vkicker	D	X	X	X		X		X	X	X	X
wiggler (map type)		X	X	X		X	X	X	X	X	X
wiggler (periodic type)	D	X	X	X		X[a]	X[a]	X[a]	X[a]	X[a]	

Tracking Code Overview

- Personal preference plays a large part in determining which code to use
- Ease of use, familiarity all reduce the chance of making a modelling error
- Using a new code often involves an initial steep learning curve

Tracking Codes I

- ‘Universal’ Lattice Codes
- Offer flexible methods for defining lattices
- Useful for optimisation and tuning of lattice parameters
- Critical components may need simulating in specialised codes (wakefields, space charge etc.)
- i.e. MAD, Elegant, LUCRETIA

Tracking Codes II

- Provide the user with a toolbox (library) which contains the needed elements and procedures
- Examples in C++, F90, Pascal
- Very flexible, can be tailored to a specific problem
- Easy to link with other codes
    - The Accelerator Markup Language / Universal Accelerator Parser Project may help in translating lattice files between codes
- i.e. BMAD, COSYINFINITY, PTC, MERLIN

Tracking Codes III

- Many codes now have a high-level interface (PYTHON, Mathematica, Matlab)
- Input/output and processing can be dealt with using built in functions
- Easy to link with other codes/implement a control system
- i.e. PyZgoubi, MAD, ELEGANT, GPT, ASTRA

Tracking Codes IV

- Programs tailored to simulate specific processes
  - i.e. Space Charge (ASTRA, GPT), CSR (CSRtrack), Ionisation cooling (ICOOL), FEL codes (Genesis, PUFFIN), Wake fields (LUCRETIA)
- Generally used to model critical sections of a lattice
- Need integrating with other codes for end-end simulations

With all tracking codes, a deep understanding of how the physical processes are implemented (and approximated) is crucial

Beam Tracking Codes

- Single particle dynamics is a mature field with many useful tools
- Codes have been benchmarked against experiments with good agreement if the accelerator model is refined enough
- For today's problems, perhaps the single particle approach is becoming less valid:
    - Beam-beam, space-charge, wakefields, IBS, Halo physics, CSR, FEL interactions, SR production
- The increase in computing speed and massively parallelised codes mean the description of the beam dynamics is becoming more and more sophisticated

Overview

- The choice of tracking code used should be driven by the underlying physics
    - What approximations are made in the underlying algorithms?
- Is symplecticity important?
- Will you need to swap data between codes?
    - Do you understand how the variables are defined?
- If speed is an issue, it may be you'll have to live with some approximations
    - You should still be aware of the approximations and try to ascertain how they will affect on your results
- Benchmarking results using codes with different approximations is ALWAYS a good idea
    - But remember, if two codes agree they could both be wrong...

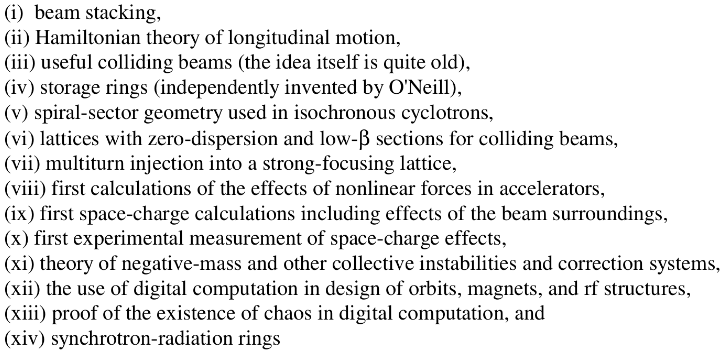
Some useful references

- The physics manual for whatever tracking code you're currently using
- 'Lie Methods for Nonlinear Dynamics with Applications to Accelerator Physics', Alex J. Dragt
  - Contains a useful introduction to numerical integration and more advanced chapters on Lie Algebra. Available from `http://www.physics.umd.edu/dsat/dsatliemethods.html`
- 'Beam Dynamics in High Energy Particle Accelerators', Andy Wolski, Imperial College Press

  (2014)
  - Very good introduction to Matrix tracking and Lie Algebra (also discusses a symplectic RK integrator)
- Most introductory accelerator text books (e.g. Lie, Wiedemann) will give a useful introduction to tracking with linear matrices
- 'Geometric Integration for Particle Accelerators', E Forest, J.Phys A:Math.Gen (39)

  2006,5321-5377
  - An ideosycratic view of the history and development of symplectic integrators
- 'O Camelot - A memoir of the MURA years', F.T. Cole
  - A personal memoir of an accelerator physicist in the 1050s and 1960s
  - Very good overview of the physics, personalities and the politics at the time

- Some achievements in the 1950s and 1960s:



(i) beam stacking,
(ii) Hamiltonian theory of longitudinal motion,
(iii) useful colliding beams (the idea itself is quite old),
(iv) storage rings (independently invented by O'Neill),
(v) spiral-sector geometry used in isochronous cyclotrons,
(vi) lattices with zero-dispersion and low-β sections for colliding beams,
(vii) multiturn injection into a strong-focusing lattice,
(viii) first calculations of the effects of nonlinear forces in accelerators,
(ix) first space-charge calculations including effects of the beam surroundings,
(x) first experimental measurement of space-charge effects,
(xi) theory of negative-mass and other collective instabilities and correction systems,
(xii) the use of digital computation in design of orbits, magnets, and rf structures,
(xiii) proof of the existence of chaos in digital computation, and
(xiv) synchrotron-radiation rings

- Many of these concepts were years (decades?) away from being practically realised