CERN/EP/DHR 78-1
8 March 1978

A FAST RUNGE KUTTA METHOD FOR
FITTING TRACKS IN A MAGNETIC FIELD

Jan Myrheim^(*) and Lars Bugge^(*)

CERN, Geneva, Switzerland

CM-P00056486

1. INTRODUCTION

The different Runge Kutta integration methods that are being used for tracking charged particles through magnetic fields, in general combine high precision with high speed [1]. These methods are also often used for fitting tracks when high precision is needed, but have then had the disadvantage of being rather slow, because they did not give derivatives of coordinates with respect to the track parameters to be fitted, i.e. the initial values in the integration.

The method described here can be used to compute the required derivatives together with the track coordinates themselves, at the expense of little extra computer time. We have used a Runge Kutta tracking routine RKPATh written by H. Wind [2] and essentially added just a few lines of FORTRAN. The routine has been tested inside a program for track fitting with vertex constraints that has been developed for the CERN SPS experiment WA7 [3], originally based on the quintic spline method [4].

We first discuss the method, and then we present the results of a few test-runs where we have compared this new Runge Kutta method and the quintic spline fit. These tests show that the two methods are quite comparable, both with respect to speed and accuracy.

(*) Visitors from Physics Institute, University of Oslo.

2. BASIC EQUATIONS

The equation of motion for a particle of mass m and charge q in a magnetic field $\vec{B}$ is

$$\frac{d\vec{p}}{dt} = q\vec{v} \times \vec{B} \quad (\text{in MKSA units}) \quad (2.1)$$

$\vec{v}$ is the velocity and

$$\vec{p} = \frac{m\vec{v}}{\sqrt{1 - \frac{v^2}{c^2}}} \quad (2.2)$$

is the momentum of the particle.

We rewrite eq. (2.1) in the form

$$u'' = \frac{q}{p} f(u, u', z) \quad (2.3)$$

with

$$u = \begin{pmatrix} x \\ y \end{pmatrix} \quad f = \begin{pmatrix} f_x \\ f_y \end{pmatrix}$$

$$u' = \frac{du}{dz} \quad u'' = \frac{d^2u}{dz^2}$$

$$f_x = \sqrt{1 + (x')^2 + (y')^2} \left(x'y'B_x - \left(1 + (x')^2 \right) B_y + y' B_z \right) \quad (2.4)$$

$$f_y = \sqrt{1 + (x')^2 + (y')^2} \left(\left(1 + (y')^2 \right) B_x - x'y' B_y - x'B_z \right).$$

The solution of eq. (2.3) is determined by the constant of motion

$$\lambda = \frac{q}{p} \quad (2.5)$$

and by the initial values at a given $z = z_o$,

$$u(z_o) = \begin{pmatrix} x_o \\ y_o \end{pmatrix}, \quad u'(z_o) = \begin{pmatrix} a \\ b \end{pmatrix}. \quad (2.6)$$

Given a set of observed coordinates on a track, the problem is to make a fit for the five variables x_o , y_o , a , b and λ . In order to have an efficient iteration procedure for the fit we want the derivatives of $u(z)$ and $u'(z)$ with respect to x_o , y_o , a , b and λ .

If $u(z)$ is the solution with initial values x_o , y_o , a , b , λ , and $(u + \Delta u)(z)$ is the solution with the perturbed initial values $x_o + \Delta x_o$, $y_o + \Delta y_o$, $a + \Delta a$, $b + \Delta b$, $\lambda + \Delta \lambda$, then we have

$$u'' = \lambda f(u, u', z) \quad (2.7)$$

$$u(z_o) = \begin{pmatrix} x_o \\ y_o \end{pmatrix} \quad u'(z_o) = \begin{pmatrix} a \\ b \end{pmatrix}$$

and

$$\begin{aligned} u'' + \Delta u'' &= (\lambda + \Delta \lambda) f(u + \Delta u, u' + \Delta u', z) \\ (u + \Delta u)(z_o) &= \begin{pmatrix} x_o + \Delta x_o \\ y_o + \Delta y_o \end{pmatrix} \\ (u' + \Delta u')(z_o) &= \begin{pmatrix} a + \Delta a \\ b + \Delta b \end{pmatrix}. \end{aligned} \quad (2.8)$$

We write

$$\lambda f(u + \Delta u, u' + \Delta u', z) \quad (2.9)$$

$$= \lambda f(u, u', z) + C(u, u', z) \Delta u + A(u, u', z) \Delta u'$$

where C and A are 2×2 matrices, for example

$$A = \lambda \frac{\partial f}{\partial u'} = \lambda \begin{pmatrix} \frac{\partial f_x}{\partial x'} & \frac{\partial f_x}{\partial y'} \\ \frac{\partial f_y}{\partial x'} & \frac{\partial f_y}{\partial y'} \end{pmatrix} \quad (2.10)$$

and we arrive at the equation for Δu ,

$$\Delta u'' = \Delta \lambda f + C \Delta u + A \Delta u' \quad (2.11)$$

with the initial conditions

$$\Delta u(z_0) = \begin{pmatrix} \Delta x_0 \\ \Delta y_0 \end{pmatrix}, \quad \Delta u'(z_0) = \begin{pmatrix} \Delta a \\ \Delta b \end{pmatrix} \quad (2.12)$$

The general solution for the perturbation Δu is

$$\Delta u(z) = \Delta \lambda L(z) + N(z) \Delta u(z_0) + M(z) \Delta u'(z_0), \quad (2.13)$$

where L is a 2×1 matrix, N and M are 2×2 matrices, and

$$L'' = f + CL + AL' \quad (2.14)$$

$$L(z_0) = 0, \quad L'(z_0) = 0$$

$$N'' = CN + AN' \quad (2.15)$$

$$N(z_0) = I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad N'(z_0) = 0$$

$$M'' = CM + AM' \quad (2.16)$$

$$M(z_0) = 0, \quad M'(z_0) = I.$$

From eq. (2.13) we read out the required derivatives,

$$\begin{aligned}\frac{\partial u(z)}{\partial \lambda} &= L(z), & \frac{\partial u'(z)}{\partial \lambda} &= L'(z) \\ \frac{\partial u(z)}{\partial u(z_o)} &= \begin{pmatrix} \frac{\partial x(z)}{\partial x_o} & \frac{\partial x(z)}{\partial y_o} \\ \frac{\partial y(z)}{\partial x_o} & \frac{\partial y(z)}{\partial y_o} \end{pmatrix} = N(z) \\ \frac{\partial u'(z)}{\partial u(z_o)} &= N'(z) \\ \frac{\partial u(z)}{\partial u'(z_o)} &= M(z) \\ \frac{\partial u'(z)}{\partial u'(z_o)} &= M'(z)\end{aligned}\tag{2.17}$$

When fitting a vertex between several tracks we need, in addition, the derivatives with respect to z_o . However, these can be computed from the above derivatives, because a solution with initial values x_o, y_o, a, b, λ at $z = z_o + \Delta z_o$ is the same as a solution with initial values

$$\begin{aligned}x_o &= a \Delta z_o \\ y_o &= b \Delta z_o \\ a &= \lambda f_x(x_o, y_o, a, b, z_o) \Delta z_o \\ b &= \lambda f_y(x_o, y_o, a, b, z_o) \Delta z_o\end{aligned}\tag{2.18}$$

and λ at z_o . Therefore,

$$\begin{aligned}
 \frac{\partial u(z)}{\partial z_o} = & -a \frac{\partial u(z)}{\partial x_o} - b \frac{\partial u(z)}{\partial y_o} \\
 & - \lambda f_x(x_o, y_o, a, b, z_o) \frac{\partial u(z)}{\partial a} \\
 & - \lambda f_y(x_o, y_o, a, b, z_o) \frac{\partial u(z)}{\partial b}
 \end{aligned} \tag{2.19}$$

and similarly for $\frac{\partial u'(z)}{\partial z_o}$.

In principle we ought to solve the eqs (2.14, 2.15 and 2.16) together with eq. (2.3). What we do in practice, however, is to define an approximation to the solution of eq. (2.3) and compute the derivatives within this approximation.

3. RUNGE KUTTA INTEGRATION

The particular Runge Kutta formula used is due to Nyström [5].

The step from z_n to $z_{n+1} = z_n + h$ is

$$u_{n+1} = u_n + hu'_n + \frac{h^2}{6} (K_1 + K_2 + K_3) \tag{3.1}$$

$$u'_{n+1} = u'_n + \frac{h}{6} (K_1 + 2K_2 + 2K_3 + K_4),$$

where

$$K_i = \lambda f_i \text{ for } i = 1, 2, 3, 4$$

$$f_1 = f(u_n, u'_n, z_n) \tag{3.2}$$

$$f_2 = f\left(u_n + \frac{h}{2} u'_n + \frac{h^2}{8} K_1, u'_n + \frac{h}{2} K_1, z_n + \frac{h}{2}\right)$$

$$f_3 = f\left(u_n + \frac{h}{2} u'_n + \frac{h^2}{8} K_1, u'_n + \frac{h}{2} K_2, z_n + \frac{h}{2}\right)$$

$$f_4 = f\left(u_n + hu'_n + \frac{h^2}{2} K_3, u'_n + h K_3, z_n + h\right).$$

Differentiating these formulae with respect to, say, x_0 , we get recursion formulae for the derivatives $\frac{\partial u(z)}{\partial x_0}$ and $\frac{\partial u'(z)}{\partial x_0}$,

$$\frac{\partial u_{n+1}}{\partial x_0} = \frac{\partial u_n}{\partial x_0} + h \frac{\partial u'_n}{\partial x_0} + \frac{h^2}{6} \frac{\partial}{\partial x_0} (K_1 + K_2 + K_3)$$

(3.3)

$$\frac{\partial u'_{n+1}}{\partial x_0} = \frac{\partial u'_n}{\partial x_0} + \frac{h}{6} \frac{\partial}{\partial x_0} (K_1 + 2K_2 + 2K_3 + K_4),$$

where

$$\begin{aligned} \frac{\partial K_1}{\partial x_0} &= C_1 \frac{\partial u_n}{\partial x_0} + A_1 \frac{\partial u'_n}{\partial x_0} \\ \frac{\partial K_2}{\partial x_0} &= C_2 \left(\frac{\partial u_n}{\partial x_0} + \frac{h}{2} \frac{\partial u'_n}{\partial x_0} + \frac{h^2}{8} \frac{\partial K_1}{\partial x_0} \right) \\ &\quad + A_2 \left(\frac{\partial u'_n}{\partial x_0} + \frac{h}{2} \frac{\partial K_1}{\partial x_0} \right) \end{aligned}$$

(3.4)

$$\begin{aligned} \frac{\partial K_3}{\partial x_0} &= C_3 \left(\frac{\partial u_n}{\partial x_0} + \frac{h}{2} \frac{\partial u'_n}{\partial x_0} + \frac{h^2}{8} \frac{\partial K_1}{\partial x_0} \right) \\ &\quad + A_3 \left(\frac{\partial u'_n}{\partial x_0} + \frac{h}{2} \frac{\partial K_2}{\partial x_0} \right) \\ \frac{\partial K_4}{\partial x_0} &= C_4 \left(\frac{\partial u_n}{\partial x_0} + h \frac{\partial u'_n}{\partial x_0} + \frac{h^2}{2} \frac{\partial K_3}{\partial x_0} \right) \\ &\quad + A_4 \left(\frac{\partial u'_n}{\partial x_0} + h \frac{\partial K_3}{\partial x_0} \right). \end{aligned}$$

We have written

$$\begin{aligned}
 A_1 &= A(u_n, u'_n, z_n) \\
 A_2 &= A\left(u_n + \frac{h}{2} u'_n + \frac{h^2}{8} K_1, u'_n + \frac{h}{2} K_1, z_n + \frac{h}{2}\right) \\
 A_3 &= A\left(u_n + \frac{h}{2} u'_n + \frac{h^2}{8} K_1, u'_n + \frac{h}{2} K_2, z_n + \frac{h}{2}\right) \\
 A_4 &= A\left(u_n + h u'_n + \frac{h^2}{2} K_3, u'_n + h K_3, z_n + h\right)
 \end{aligned} \tag{3.5}$$

and similarly for C_1, C_2, C_3 and C_4 .

For the derivatives with respect to y_0 , a and b we get exactly similar formulae. When differentiating with respect to λ , we get similar formulae except for one extra term f_i in the formula for K_i , thus

$$\begin{aligned}
 \frac{\partial K_1}{\partial \lambda} &= f_1 + C_1 \frac{\partial u_n}{\partial \lambda} + A_1 \frac{\partial u'_n}{\partial \lambda} \\
 \frac{\partial K_2}{\partial \lambda} &= f_2 + C_2 \left(\frac{\partial u_n}{\partial \lambda} + \frac{h}{2} \frac{\partial u'_n}{\partial \lambda} + \frac{h^2}{8} \frac{\partial K_1}{\partial \lambda} \right) \\
 &\quad + A_2 \left(\frac{\partial u'_n}{\partial \lambda} + \frac{h}{2} \frac{\partial K_1}{\partial \lambda} \right)
 \end{aligned} \tag{3.6}$$

etc.

4. A USEFUL APPROXIMATION

We will assume that the x and y partial derivatives of the magnetic field components are all zero along the track,

$$\frac{\partial B_i}{\partial x} = \frac{\partial B_i}{\partial y} = 0 \text{ for } i = x, y, z \tag{4.1}$$

implying that

$$C = \lambda \frac{\partial f}{\partial u} = 0. \quad (4.2)$$

This is certainly a good approximation in many cases, and it simplifies considerably the above formulae.

Thus, if we write out in detail eq. (3.6) we now get

$$\begin{aligned} \frac{\partial K_1}{\partial \lambda} &= f_1 + A_1 \frac{\partial u'_n}{\partial \lambda} \\ \frac{\partial K_2}{\partial \lambda} &= f_2 + \frac{h}{2} A_2 f_1 + A_2 \left(I + \frac{h}{2} A_1 \right) \frac{\partial u'_n}{\partial \lambda} \\ \frac{\partial K_3}{\partial \lambda} &= f_3 + \frac{h}{2} A_3 \left(f_2 + \frac{h}{2} A_2 f_1 \right) \\ &\quad + A_3 \left(I + \frac{h}{2} A_2 \left(I + \frac{h}{2} A_1 \right) \right) \frac{\partial u'_n}{\partial \lambda} \\ \frac{\partial K_4}{\partial \lambda} &= f_4 + h A_4 \left(f_3 + \frac{h}{2} A_3 \left(f_2 + \frac{h}{2} A_2 f_1 \right) \right) \\ &\quad + A_4 \left(I + h A_3 \left(I + \frac{h}{2} A_2 \left(I + \frac{h}{2} A_1 \right) \right) \right) \frac{\partial u'_n}{\partial \lambda}. \end{aligned} \quad (4.3)$$

The recursion step for $\partial u / \partial \lambda$ and $\partial u' / \partial \lambda$ now takes the following form:

$$\begin{aligned} \frac{\partial u_{n+1}}{\partial \lambda} &= \frac{\partial u_n}{\partial \lambda} + h \frac{\partial u'_n}{\partial \lambda} + \frac{h^2}{6} \frac{\partial}{\partial \lambda} \left(K_1 + K_2 + K_3 \right) \\ &= \frac{\partial u_n}{\partial \lambda} + h \frac{\partial u'_n}{\partial \lambda} \\ &\quad + \frac{h}{3} \left\{ \frac{h}{2} A_3 + \frac{h}{2} A_1 \right. \\ &\quad \quad \left. + \left(I + \frac{h}{2} A_3 \right) \frac{h}{2} A_2 \left(I + \frac{h}{2} A_1 \right) \right\} \frac{\partial u'_n}{\partial \lambda} \\ &\quad + \frac{h^2}{6} \left\{ f_3 + f_1 + \left(I + \frac{h}{2} A_3 \right) \left(f_2 + \frac{h}{2} A_2 f_1 \right) \right\} \end{aligned} \quad (4.4)$$

$$\begin{aligned}
\frac{\partial u'_{n+1}}{\partial \lambda} &= \frac{\partial u'_n}{\partial \lambda} + \frac{h}{6} \frac{\partial}{\partial \lambda} \left(K_1 + 2K_2 + 2K_3 + K_4 \right) \\
&= \left\{ \frac{1}{3} \left(I + \frac{h}{2} A_4 + \frac{h}{2} A_1 \right) \right. \\
&\quad + \frac{2}{3} \left(I + \left(I + \frac{h}{2} A_4 \right) \frac{h}{2} A_3 \right) \left(I + \frac{h}{2} A_2 \left(I + \frac{h}{2} A_1 \right) \right) \left. \right\} \frac{\partial u'_n}{\partial \lambda} \\
&\quad + \frac{h}{6} \left\{ f_4 + f_1 + 2 \left(I + \frac{h}{2} A_4 \right) f_3 \right. \\
&\quad + \left. 2 \left(I + \left(I + \frac{h}{2} A_4 \right) \frac{h}{2} A_3 \right) \left(f_2 + \frac{h}{2} A_2 f_1 \right) \right\}.
\end{aligned} \tag{4.5}$$

The formulae for the derivatives with respect to x_o , y_o , a and b are similar to eqs (4.4) and (4.5), with the only difference that they do not contain the terms with f_1 , f_2 , f_3 and f_4 .

5. TEST RESULTS

We have made some test runs using alternatively the Runge Kutta or the quintic spline fit as part of the off-line analysis chain of the elastic scattering experiment WA7 [6]. The tests were made both with Monte-Carlo generated elastic events and with real non-elastic events, fitting two outgoing tracks with vertex constraints. On our Monte-Carlo elastic data we also imposed the momentum conservation constraints. All data relate to an incident π^- beam at 20 GeV/c.

Running on Monte Carlo data we used the Monte-Carlo generated coordinates with no generated errors; this was therefore a test of the accuracy of the two methods. We used weights corresponding to measurement errors of 0.5 mm in all coordinates and the first iteration was started from perturbed values for the Monte-Carlo generated vertex position and momenta. We added to the correct values perturbations of ± 3 cm in x and y of the vertex, ± 15 cm in z of vertex, ± 0.05 in dx/dz and dy/dz of the two tracks at vertex, and $\pm 20\%$ in the momenta.

Running on real data we have used weights corresponding to errors of 2 mm in single detector planes. The question of whether or not these weights are correct, should not be important for the comparison between the two methods. The starting point for the first iteration on a real event was the vertex position and momenta given as output from the pattern recognition program.

We incorporated a convergence test, which was that the "step squared" in one iteration should be less than 0.01. Let Δ_i be the weighted residual in the i -th measured coordinate after a given number of iterations, then at that stage in the fitting process the approximate value for χ^2 is, assuming independent measurements,

$$\chi^2 = \sum_i \Delta_i^2 . \quad (5.1)$$

If after the next iteration, the weighted residuals are Δ'_i , we define the step length by

$$(\text{step})^2 = \sum_i (\Delta'_i - \Delta_i)^2 . \quad (5.2)$$

This is the most natural measure of how much the parameters to be fitted are changed during one iteration.

In order to get an idea of the relative speed and precision of the two methods, we used them in such a way that they computed the magnetic field at nearly the same points. That is, we used steps in z of 25 cm for the Runge Kutta integration, and 12.5 cm for the quintic spline fit.

Of 500 Monte-Carlo generated elastic events all converged from the perturbed event variables back to the correct values, giving a χ^2 of effectively zero. Out of a sample of 1000 real triggers with altogether 1113 candidates for two-prong vertices, many vertices failed to give a fit, for different reasons. The Runge Kutta method finally made a fit for 813 vertices, whereas the quintic spline fitted 798 vertices. We do not consider this 2% difference to be very significant, however,

since many vertices have an ill-defined z-coordinate because the two tracks are nearly parallel at the vertex, and these vertices may cause convergence problems.

6. EXECUTION TIMES

Table 1 summarizes the results of our timing on an IBM 370 computer, under the different conditions.

Table 1

Execution times (ms/event)

	Runge Kutta	Quintic spline
Monte-Carlo data (6 iterations, no convergence test)	146	138
Monte-Carlo data (stop after convergence)	121	123
Real data (ms/input vertex, ≤ 8 iterations, with convergence test)	126	123
Real data (same run, ms/fitted vertex)	173	172

We should add a few remarks. We have used a magnetic field routine requiring 0.126 ms for one call, and for 6 iterations on Monte-Carlo data the total time spent in computing the field was 34 ms/event for either program. In addition, in the quintic spline fit, before the first iteration a Runge Kutta tracking was done in order to define starting values for coordinates and derivatives, and this tracking required about 15 ms/event, including 11 ms/event for computing field values.

Thus, both methods could be speeded up somewhat by a more efficient representation of the magnetic field, in discrete planes only, and by not computing the field in every iteration. Also, a more efficient method for defining first approximations could save some time for the quintic spline fit.

7. CONVERGENCE PROPERTIES

We have also compared the convergence of the two methods, and the results are shown in figs 1 to 3 for Monte-Carlo data and in figs 4 to 6 for real data.

Figs 1 and 4 show the mean of the logarithm (to base 10) of the approximation to χ^2 , after 0, 1, 2, ... iterations. Figs 2 and 5 show the mean of the logarithm of the "step squared" in iteration 1, 2, 3, ... Finally, figs 3 and 6 show the number of events left after 0, 1, 2, ... iterations, when the convergence test is applied. On real data, events are sometimes cut out during the fitting, not because they have converged, but because they diverge in some way or other (tracks pass outside boundaries of the magnetic field, etc.). Thus the convergence test starts cutting the number of events only in the fourth iteration.

8. CONCLUSION

From the test we have done, using fixed steps through the magnetic field, it appears that the Runge Kutta method spends somewhat more time per iteration, but on the other hand it is more precise and converges faster than the quintic spline method. If both methods are optimised to give a certain precision with a minimum of time spent in computing values of the magnetic field, one method or the other might be the most efficient, depending on which one of the effects mentioned turns out to be the most important one.

We therefore conclude that the present Runge Kutta method can compete with the quintic spline fit, not only in precision, but also in speed.

REFERENCES

- [1] M.G. Shiel, A Comparison of Routines Used to Follow Particle Trajectories through Magnetic Fields, Internal Note CERN/DD 75-22.
- [2] CERN Program Library, algorithm S4101.
- [3] R.K. Böck, Quintic Spline Fit with Inter-Track Constraints, WA7 Internal Note ELAS 77.
- [4] H. Wind, Momentum Analysis by Using a Quintic Spline Model for the Track, Nucl. Inst. and Meth. 115 (1974) 431.
- [5] National Bureau of Standards Handbook, procedure 25.5.20.
- [6] Study of Exclusive Hadronic Processes at large p_T , CERN/SPSC 74-28 (P-9) and CERN/SPSC 74-49 (P-9, Add. 1) March/April 1974.

FIGURE CAPTIONS

Fig. 1 Mean of $\log \chi^2$ before iteration 1, 2, 3, ... for 500 Monte-Carlo events.

Fig. 2 Mean of $\log (\text{step}^2)$ in iteration 1, 2, 3, ... for the same Monte-Carlo events.

Fig. 3 Number of events passed on from iteration $n-1$ to iteration n , $n = 1, 2, 3, \dots$, Monte-Carlo data.

Fig. 4-6 The same as fig. 1-3, but for real data.

Fig. 1

Fig. 2

Fig. 3

Fig. 4

Fig. 6

Fig. 5