TRIUMF Beam Physics Note
TRI-BN-18-09
July, 2018, revised Aug. 2018

Runge-Kutta Customization for TRANSOPTR

R. Baartman

TRIUMF

Abstract: Out-of-the-box Runge-Kutta routines with adaptive step size use internal techniques to detect truncation error. As some of the integrated parameters are oscillatory and pass through zero, a relative error calculation is inappropriate. This can be overcome by using the known physics of the integrated orbits to set scales against which to compare the calculated errors. The implementation of this technique for TRANSOPTR is described.

4004 Wesbrook Mall, Vancouver, B.C. Canada V6T 2A3 Tel: 604 222-1047 Fax: 604 222-1074 www.triumf.ca

1 Introduction

In space charge mode, TRANSOPTR integrates the σ-matrix as well as the transfer matrix through the elements. This allows not only space charge, but also any element whose Hamiltonian is known. Lately, because of this feature, we have added linear accelerators, soft-edge solenoids, and many other elements[1].

Since 1984, the integration engine is a Runge-Kutta (RK) type, since its self-starting characteristics make it ideal for orbit calculation. The code used was a public domain code received from UBC computing, named RKC, written by Carolyn Moore, April 1971. It automatically adapts the step size to attain a user-specified accuracy. This FORTRAN routine was translated from ALGOL code written by Jan Christiansen[2]. The particular algorithm is fourth order and due to Merson[3].

Comparing calculated beam envelopes against profile measurements is not very demanding regarding accuracy. However, accuracy of order $< 10^{-5}$ is required when the code is called on to optimize a beamline design or to find the optimum tune for an as-built beamline. Over decades of use, RKC has proven to give such accuracy despite being only single-precision and has shown itself to be impressively robust.

But optimizing tunes for the electron linac for example becomes difficult becuase to attain 5-figure accuracy per step requires steps to be as short as 1 mm and therefore typically 10^3 RK steps through a 1-metre linac. Step-to-step errors can accumulate linearly in the worst cases, bringing overall accuracy to the fraction of a percent level.

For this reason, we investigated how to finetune the RK routine for the particular problem at hand, customizing it for the optics case.

2 RKC "under the hood"

RKC solves the system of N coupled equations

$$\frac{dy_i}{ds} = f_i(s, y_1, y_2, ..., y_N) \tag{1}$$

It also derives an upper bound to the error (Δy_i), and compares it to the desired user-specified error (ϵ). This specified error is supposed to be a relative error, so it tests whether

$$|\Delta y_i| < \epsilon |y_i|. \tag{2}$$

If this condition is met, integration continues with same step size. If it is met by a factor 32, stepsize is doubled. If it is not met, step size is halved and the step re-taken.

Clearly, this procedure is invalid any time the solution y_i passes through zero. `RKC` recognizes this, so if $|y_i|$ is small, less than 0.001, it reverts to absolute comparison:

$$|\Delta y_i| < \epsilon \text{ for } |y_i| < 0.001. \tag{3}$$

This is arbitrary at best.

To focus discussion, think of the 2D phase space case where

$$\begin{pmatrix} \sigma'_{11} & \sigma'_{12} \\ \sigma'_{12} & \sigma'_{22} \end{pmatrix} = \begin{pmatrix} 2\sigma_{12} & \sigma_{22} - K\sigma_{11} \\ \sigma_{22} - K\sigma_{11} & -2K\sigma_{12} \end{pmatrix}, \tag{4}$$

where $K = K(s)$ is the local focal strength. So for example, if using metres and radians for resp. beam extent and divergence, the σ-matrix elements for a mm, mrad sized beam, being the squares of the actual sizes, are of order 10^{-6}. This would invalidate the `RKC` error evaluation technique. `TRANSOPTR` uses cm as internal units, so this explains why it worked fairly well: most charged particle beams are mm to cm in size.

3 The Fix

3.1 Scales

`TRANSOPTR` integrates more than just the σ-matrix; it also integrates the transfer matrix, and the first order coordinates when needed (e.g. time and energy for linear accelerators): a total of 78 ODEs in all ($36 + 36 + 6$). To fix the problem, the errors in the 78 dependent variables y_i should be compared against a scale of their expected order or range. In other words, we want to use the criterion

$$|\Delta y_i| < \epsilon |s_i|, \tag{5}$$

where s_i is a vector of scales. This vector must be locally determined.

3.1.1 σ-matrix

For the full 6D σ-matrix, the fix is surprisingly simple. All elements σ_{ij} are of order $\sqrt{\sigma_{ii}\sigma_{jj}}$. (Also trivially true for the diagonal elements.) This follows from the fact that normalized off-diagonal elements are always between $+1$ and -1, and their

normalization factor is $\sqrt{\sigma_{ii}\sigma_{jj}}$. Only the off-diagonal elements pass through zero, and the diagonal elements are positive definite. So the fix is to replace eqn. 2 with

$$|\Delta\sigma_{ij}| < \epsilon\sqrt{\sigma_{ii}\sigma_{jj}}. \tag{6}$$

3.1.2 Transfer matrix

As `TRANSOPTR` uses only canonical variables when integrating, the diagonal elements of the transfer matrix are usually of order 1. In exceptional cases of large magnification m, they may be as large as m, so the scale is set to the larger of 1 and the element itself. One might think this penalizes the adjacent diagonal element which is of order $1/m$ and will be set to a scale of 1, but this does not matter because the RK step size will be set to that needed by the diagonal element that is of order m.

Given a typical focal or element length L, the off-diagonal elements R_{ij} are of order L for $i < j$ and $1/L$ for $i > j$.

$$\frac{\Delta R_{ij}}{\epsilon} < \begin{cases} \max\{1, R_{ii}\} & i = j \\ L & i < j \\ 1/L & i > j \end{cases} \tag{7}$$

3.1.3 First order (row 13)

The first order integration of 4 positions and momenta is not yet implemented; only energy and time coordinates are calculated, for example in devices that accelerate. The scales are simple: the energy scale is just the energy itself. The time coordinate is actually ct, and so scale is c/ω; the rf wavelength over 2π.

3.2 Implementation

As is typical for RK codes, `RKC` has three nested loops. The outer is the stepper, the middle is the 5-part RK algorithm, and the inner is executed N times, once for each y_i. Between the first two loops, i.e. at every step, a call is made to `SCALER` (see appendix), with 3 arguments: (1) the integration length Δs. The length scale L is the largest of Δs, and the matrix elements R_{ij} where $i = 1, 3, 5; j = 2, 4, 6$. Similarly, an appropriate inverse length scale can be found by comparing $1/L$ with R_{ij} where $i = 2, 4, 6; j = 1, 3, 5$. Argument 2 is the vector y_i, argument 3 is the output vector s_i. This vector is used to scale ϵ and the result compared with Δy_i as in eqn. 5 to determine whether the step should be halved and re-taken, or doubled (or unchanged) and continued.

3.3 Other RK Engines

Currently there are 4 RK engines available in `TRANSOPTR`. They are chosen by including the common block `RKKIND` into the system subroutine `sy.f`. This common block has just one integer element. If this integer is zero, `RKC` will be used. This is the default. The others are as follows:

0. `RKC`: Merson type.
1. `RKGS`: Gill type.
2. `RKFS` or `RKF45`: Kutta-Fehlberg type.
3. `RKCK`: Cash-Karp, from Numerical Recipes.

All 4 have now been modified to use the scaling routine and eqn. 5.

3.3.1 `RKGS`

As stated in [4], "`RKGS` is a routine from the SSP collection which is a widely distributed program library prepared by IBM". This was also widely used at TRIUMF in the 70s, for example for cyclotron orbit codes like `CYCLOPS` and `GOBLIN`, though there, the step size could be no smaller than the magnetic field survey grid size. It uses a very simple technique for estimating error. Calculations are performed for one whole step and two half steps, and the difference taken. The error estimate of the two smaller steps is this difference divided by $2^r - 1$, where r is the order of the calculation[5, eqn. 32]. In `RKGS`, $r = 4$, so this factor is 15. `RKGS` allows the user to provide weights for the different y_i's. My modification is to replace this weighting system with the `scaler` routine. For equal ϵ, `RKGS` uses a similar number of function evaluations as `RKC`, but it seems to be slightly less accurate.

3.3.2 `RKFS`

The Fehlberg routine[6] `RKFS` simultaneously runs a fourth and a fifth order integrator, and determines error by comparing the results of the two. It is capable of finding optimum step sizes; not simply doubling and halving. The original code by Watts and Shampine[4] dealt with zero crossings by asking for both a relative error (ϵ_r) and an absolute error (ϵ_a). The criterion used was

$$|\Delta y_i| < \epsilon_r \overline{y_i} + \epsilon_a, \tag{8}$$

where $\overline{y_i}$ is the average of $|y_i|$ at the start and the end of the step. This was commented out in favour of the scheme (5) using `scaler` as outlined for `RKC`. In

TRANSOPTR, RKFS works apparently as well as or slightly better than RKC. This is judged by the accuracy attained versus number of function evaluations.

3.3.3 RKCK

The Cash-Karp routine from Numerical Recipes[8] is a Fehlberg approach with slightly more claimed efficiency[7]. It has the nice feature that it already uses a s_i scale vector `yscal(i)` explicitly. The integrator, ODEINT, originally sets this to $|y_i| + h|f_i|$ where h is the step size. This is a "trick" to avoid zero-crossing issues. But it is a simple matter to replace this statement with the condition (5) derived from the `scaler` routine. This routine performs as well as RKFS.

```
      subroutine scaler(SL,sx,sxs)
c SL is input integration length, sx is the 13x8 matrix (input),
c     sxs is output 13x8 matrix of scales
      COMMON/axezrf/omoc,dum1(2),dum2(9999),dum3(9999),dum4(9999),
     $     dum5,idum6,iaxezrf,spphase
c The RK routines adjust stepsize by estimates of errors in each of the 78 ODEs.
c SCALER calculates how to scale those errors so they can be compared to the tolerance fairly.
      dimension sx(13,6),sxs(13,6)
c Sigma matrix elements. Need absolute values because insufficient step size often results
c     in negative diagonal elements even though theoretically they must be positive definite.
      do 1 i=1,6
      do 1 j=1,6
 1       sxs(i,j)=sqrt(abs(sx(i,i)*sx(j,j)))
      dum=0.
c SL is a typical focusing length scale, WL a typical focal strength.
      do 13 i=7,12,2
      do 13 j=2,6,2
 13       dum=dum+sx(i,j)**2
          QL=amax1(sqrt(dum),abs(sl))
      dum=0.
      do 14 i=8,12,2
      do 14 j=1,6,2
 14       dum=dum+sx(i,j)**2
          WL=amax1(sqrt(dum),1./abs(ql))
c Dimensionful transfer matrix elements.
      do 3 i=7,12,2
      do 3 j=2,6,2
 3       sxs(i,j)=QL
      do 4 i=8,12,2
      do 4 j=1,6,2
 4       sxs(i,j)=WL
c Dimensionless elements of transfer matrix. In cases where they are small, they scale to 1.
c     In cases where large, like when magnification is large, they scale on themselves.
      do 5 i=7,12,2
      do 5 j=1,6,2
 5       sxs(i,j)=amax1(1.,abs(sx(i,j)))
      do 6 i=8,12,2
      do 6 j=2,6,2
 6       sxs(i,j)=amax1(1.,abs(sx(i,j)))
c The first order (row 13) are special cases.
      do 2 j=1,4
 2       sxs(13,j)=1.
         sxs(13,5)=1.
         if(omoc.ne.0.)sxs(13,5)=1./omoc
         sxs(13,6)=sx(13,6)
      return
      end
```

It may be helpful to know how the matrices map to the y_i, so I've prepared a table.

$$\begin{pmatrix} \sigma_{1,1} & \sigma_{1,2} & \sigma_{1,3} & \sigma_{1,4} & \sigma_{1,5} & \sigma_{1,6} \\ \sigma_{2,1} & \sigma_{2,2} & \sigma_{2,3} & \sigma_{2,4} & \sigma_{2,5} & \sigma_{2,6} \\ \sigma_{3,1} & \sigma_{3,2} & \sigma_{3,3} & \sigma_{3,4} & \sigma_{3,5} & \sigma_{3,6} \\ \sigma_{4,1} & \sigma_{4,2} & \sigma_{4,3} & \sigma_{4,4} & \sigma_{4,5} & \sigma_{4,6} \\ \sigma_{5,1} & \sigma_{5,2} & \sigma_{5,3} & \sigma_{5,4} & \sigma_{5,5} & \sigma_{5,6} \\ \sigma_{6,1} & \sigma_{6,2} & \sigma_{6,3} & \sigma_{6,4} & \sigma_{6,5} & \sigma_{6,6} \\ R_{1,1} & R_{1,2} & R_{1,3} & R_{1,4} & R_{1,5} & R_{1,6} \\ R_{2,1} & R_{2,2} & R_{2,3} & R_{2,4} & R_{2,5} & R_{2,6} \\ R_{3,1} & R_{3,2} & R_{3,3} & R_{3,4} & R_{3,5} & R_{3,6} \\ R_{4,1} & R_{4,2} & R_{4,3} & R_{4,4} & R_{4,5} & R_{4,6} \\ R_{5,1} & R_{5,2} & R_{5,3} & R_{5,4} & R_{5,5} & R_{5,6} \\ R_{6,1} & R_{6,2} & R_{6,3} & R_{6,4} & R_{6,5} & R_{6,6} \\ 0 & 0 & 0 & 0 & ct & \gamma - 1 \end{pmatrix} = \begin{pmatrix} y_1 & y_{14} & y_{27} & y_{40} & y_{53} & y_{66} \\ y_2 & y_{15} & y_{28} & y_{41} & y_{54} & y_{67} \\ y_3 & y_{16} & y_{29} & y_{42} & y_{55} & y_{68} \\ y_4 & y_{17} & y_{30} & y_{43} & y_{56} & y_{69} \\ y_5 & y_{18} & y_{31} & y_{44} & y_{57} & y_{70} \\ y_6 & y_{19} & y_{32} & y_{45} & y_{58} & y_{71} \\ y_7 & y_{20} & y_{33} & y_{46} & y_{59} & y_{72} \\ y_8 & y_{21} & y_{34} & y_{47} & y_{60} & y_{73} \\ y_9 & y_{22} & y_{35} & y_{48} & y_{61} & y_{74} \\ y_{10} & y_{23} & y_{36} & y_{49} & y_{62} & y_{75} \\ y_{11} & y_{24} & y_{37} & y_{50} & y_{63} & y_{76} \\ y_{12} & y_{25} & y_{38} & y_{51} & y_{64} & y_{77} \\ y_{13} & y_{26} & y_{39} & y_{52} & y_{65} & y_{78} \end{pmatrix} \tag{9}$$

References

[1] R. Baartman, "`TRANSOPTR`: Changes since 1984," TRIUMF, Tech. Rep. TRI-BN-16-06, 2016.

[2] J. Christiansen, "Numerical solution of ordinary simultaneous differential equations of the 1st order using a method for automatic step change," *Numer. Math.*, vol. 14, no. 4, pp. 317–324, Mar. 1970. [Online]. Available: http://dx.doi.org/10.1007/BF02165587

[3] R. Merson, "An operational method for the study of integration processes," in *Proc. Symp. Data Processing*. Weapons Research Establishment, Salisbury, S. Australia, 1957, pp. 1–25.

[4] H. Watts and L. Shampine, "RKF45 Runge-Kutta-Fehlberg ODE Solver," Technical report, Albuquerque, New Mexico, Tech. Rep., 2005.

[5] A. Ralston and H. S. Wilf, *Mathematical methods for digital computers, Vol. 1*. Wiley & Sons, Incorporated, 1960.

[6] E. Fehlberg, "Low-order classical runge-kutta formulas with stepsize control and their application to some heat transfer problems," 1969.

[7] J. R. Cash and A. H. Karp, "A variable order runge-kutta method for initial value problems with rapidly varying right-hand sides," *ACM Trans. Math. Softw.*, vol. 16, no. 3, pp. 201–222, Sep. 1990. [Online]. Available: http://doi.acm.org/10.1145/79505.79507

[8] W. H. Press, *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.