The COSY 8th Order Runge Kutta Integrator

K. Makino
Department of Physics
University of Illinois at Urbana-Champaign
Urbana, IL 61801-3080

March 14, 2002

Abstract

We studied simple example problems to compare the performance of the COSY 8th order Runge Kutta integrator RK, and two of fourth order Runge Kutta integrators; RK4 with fixed step size, and RK4S with automatic step size control. The resulting low accuracy by the fourth order integrators convinces us to urge to use high accuracy integrators for modern computational needs.

1 The COSY Eighth Order Runge Kutta Integrator

The COSY 8th order Runge Kutta integrator RK was originally written in Fortran77 by Ingolf Kübler in 1986 [1]. The code was translated to the COSY language by Martin Berz in 1990 to be easily accessed in COSY Infinity [2]. It was particularly important that the integration can be carried out not only for real numbers but also for Differential Algebraic vectors to support the full power of the Differential Algebraic technique in COSY Infinity for various beam physics computations, including Taylor transfer maps and efficient 3D field computations. For this special need, a very accurate and robust integrator was demanded. The current COSY integrator RK has been used for more than ten years for numerous beam physics computations with excellent accuracy and robustness.

So often, fourth order Runge Kutta integrators are used widely for various numerical problems. When the error is controlled with the combination of automatic step size, the accuracy can be improved. However, a blind usage of fourth order Runge Kutta integrators would lead to a false answer for complicated systems.

In this note, we study a few simple example problems to assess the accuracy of the COSY 8th order integrator RK and two fourth order integrators RK4 and RK4S. RK4 keeps a fixed step size, and RK4S has automatic step size control. It is not surprising that the COSY integrator RK has much higher accuracy. More importantly, the results show clearly that the widely spread fourth order Runge Kutta integrators are not accurate enough for many modern computational needs. To assist researchers in the beam physics community, the COSY integrator RK is now directly translated from the COSY language written code to Fortran so that the routines can be imported to any Fortran code easily. The test programs are added in the package "testRK" as well as the codes for RK4 and RK4S.

2 Fourth Order Runge Kutta Integrators

We use the standard fourth order Runge Kutta algorithm for the integrator RK4 and RK4S [3, 4].

$$\begin{aligned}
y_{n+1} &= y_n + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 \\
k_1 &= h\, f(x_n, y_n) \\
k_2 &= h\, f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1) \\
k_3 &= h\, f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2) \\
k_4 &= h\, f(x_n + h, y_n + k_3).
\end{aligned}$$

RK4 follows the algorithm with fixed step size. RK4S combines the idea of adaptive step size control discussed in [4] with the current step size control algorithm in the COSY integrator RK.

At each time step in RK4S, we make two estimates $y^{(1)}$ and $y^{(2)}$, where $y^{(1)}$ is an estimate by one step of the size $2h$, and $y^{(2)}$ is an estimate by two steps of the size h. Because the error of the fourth order algorithm is $O(h^5)$, the difference $\Delta \equiv y^{(2)} - y^{(1)}$ provides a fifth order estimate for the solution as

$$y(x + 2h) = y^{(2)} + \frac{\Delta}{15} + O(h^6)$$

[4]. Instead of using the exact step size control algorithm in [4], we use the current step size control algorithm in the COSY integrator RK. Since we don't carry any higher order information in RK4S, we utilize Δ for the error estimate.

The behavior of the step size change is one interesting issue to study for numerical integration, and all the successful computations have a similarity in the pattern of step size change. Since RK4S adopts the same step size control algorithm of RK, those two integrators have a similar step size development.

3 Numerical Examples

We applied the three Runge Kutta integrators to simple examples with different dimensionality and properties. The first one is a one dimensional integral, the solution of which is precisely known. The second one is a two dimensional problem with a closed orbit solution. The last one is a three dimensional problem known to have chaotic behavior, and is thus challenging for any numerical integrator.

3.1 One Dimensional Integral with Known Answer

The first example problem is the integral

$$4\int_0^1 \sqrt{1-t^2}dt,$$

where the solution is $\pi \simeq 3.14159265358979...$ The computational results are summarized in Table 1. The COSY integrator RK performs well as the 8th order method, but both RK4 and RK4S give somewhat shocking results for this simple one dimensional problem.

	Numerical Answer	Error Estimate
RK4	3.14143024919	Unavailable
RK4S	3.14455379964	2.7778011E-3
RK	3.14159262801	5.9354652E-7

Table 1: One dimensional integral $4\int_0^1 \sqrt{1-t^2}dt = \pi$.

3.2 The Volterra Equations

The next example is a system of two dimensional ordinary differential equations. We study the Volterra equations, which describe two conflicting populations, and the solution trajectories are known to form closed orbits if the values of initial condition are positive. We use the following equations and initial conditions

$$\frac{dx_1}{dt} = 2x_1(1-x_2)$$
$$\frac{dx_2}{dt} = -x_2(1-x_1)$$
$$x_1(0) = 1, \; x_2(0) = 3.$$

The solution trajectory follows the constraint

$$x_1 x_2^2 e^{-x_1-2x_2} = \text{Constant} = 0.00820693768999064 5,$$

and the cycle is about $T = 5.488138468035$.

We carried the integration using RK4, RK4S and RK from $t = 0$ to $t = T$. Table 2 summarizes the results, showing the final positions $x_1(T)$ and $x_2(T)$, the computed error estimates, and the values of $x_1x_2^2e^{-x_1-2x_2}$ at $t = T$ to show how close the numerical results are to the true solution. The accuracy of the COSY integrator RK again shows significant superiority compared to the fourth order integrators.

	$x_1(T)$	$x_2(T)$	Error	$x_1x_2^2e^{-x_1-2x_2}$ at T
RK4	0.99965947825	2.99983170872	Unavailable	0.0082087789360943
RK4S	1.00000424005	3.00000143890	5.0705462E-4	0.0082069219446463
RK	1.00000000014	2.99999999987	8.3244496E-9	0.0082069376914314

Table 2: Integration of the Volterra equations for about one cycle T.

3.3 The Lorenz System

The last example is a system of three dimensional ordinary differential equations, the Lorenz system, which describes a model of unpredictable turbulent flows in fluid dynamics. The system is very sensitive to the initial condition, and more specifically its behavior is known to be chaotic, hence it presents a challenge to any numerical integrator. We use the following equations and initial conditions

$$\begin{aligned}\frac{dx_1}{dt} &= 10(x_2 - x_1)\\ \frac{dx_2}{dt} &= x_1(28 - x_3) - x_2\\ \frac{dx_2}{dt} &= x_1x_2 - \frac{8}{3}x_3\\ x_1(0) = 15,&\ x_2(0) = 15,\ x_3(0) = 36.\end{aligned}$$

We carried the integration using RK4, RK4S and RK from $t = 0$ to $t = T = 20$. Table 3 summarizes the results, showing the final positions $x_1(T)$, $x_2(T)$ and $x_3(T)$, and the computed error estimates, which are not validated. The final positions differ so much between the three integrators, and even the error estimates seem to not make much sense. For this problem, some illustrations are helpful to analyze the performance.

	$x_1(T)$	$x_2(T)$	$x_3(T)$	Error
RK4	2.20560898813	1.03043280578	22.2823992447	Unavailable
RK4S	-3.38810890144	0.796427735778	27.5820328868	1.2784637E-2
RK	14.3095004639	9.59194363188	39.0397302363	1.2144947E-7

Table 3: Integration of the Lorenz system from $t = 0$ to $t = T = 20$.

Figure 1 monitors the x_1 position along time t. Until $t \sim 10$, all the integrators keep essentially the same x_1 position, then RK4 deviates, and at $t \sim 15$, RK4S also deviates. The x_2 and x_3 positions behave similarly. Once a deviation happens, it develops to very different positions. Figure 2 shows the trajectories computed by RK, RK4 and RK4S.

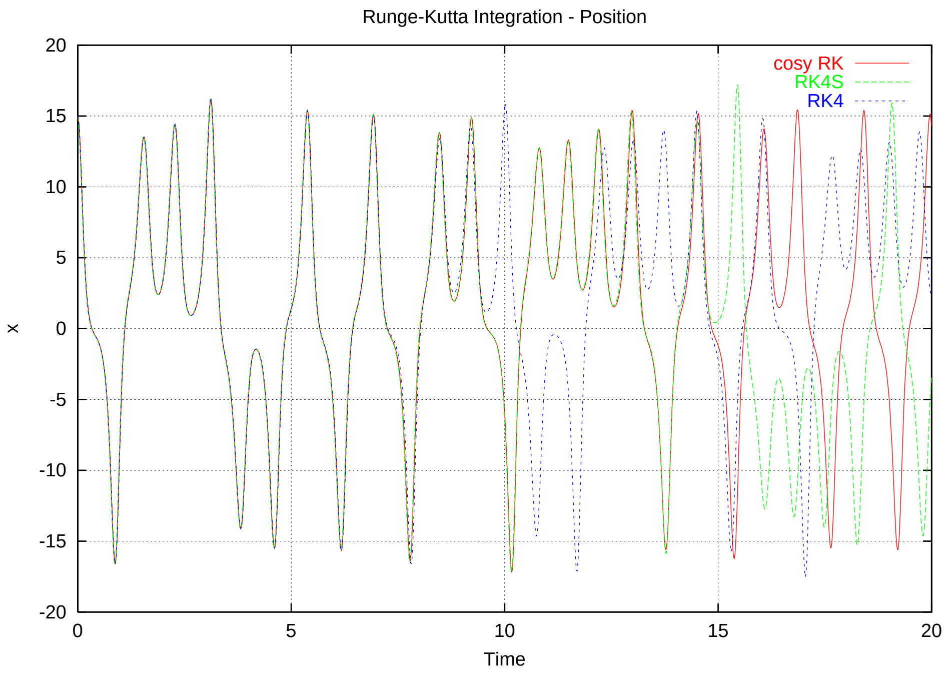

Figure 1: The position x_1 of the Lorenz system computed by RK, RK4S and RK4.

Of course, it is important to estimate the necessary computational expense. Table 4 lists a CPU time comparison. The numbers are normalized to the CPU time for RK. The result shows that the fourth order algorithms don't have any advantage besides their obvious simplicity in implementation.

	CPU time
RK4	0.689
RK4S	0.904
RK	1.000

Table 4: CPU time for the integration of the Lorenz system. The numbers are normalized to the CPU time for RK.

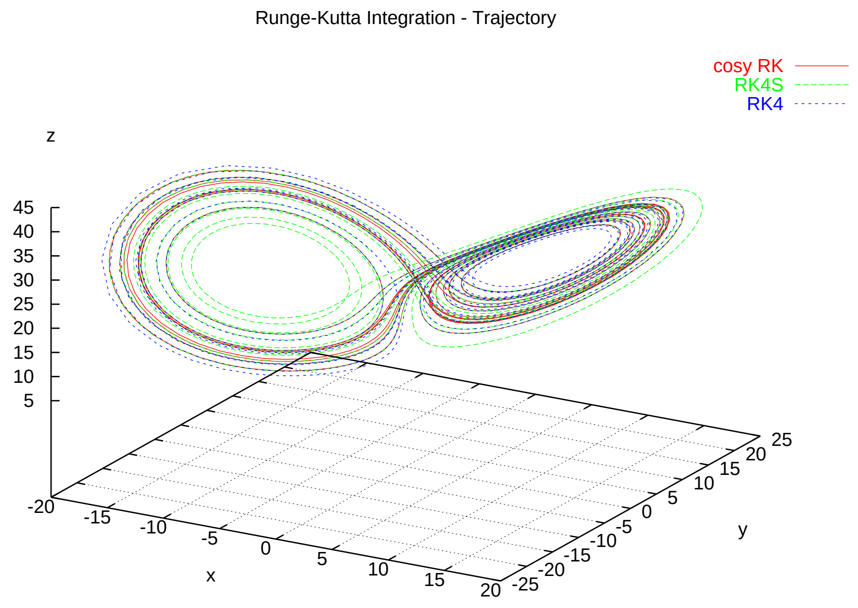

Figure 2: The trajectories of the Lorenz system computed by RK, RK4S and RK4.

4 Conclusion

We provided a performance comparison between the COSY 8th order Runge Kutta integrator RK and two fourth order Runge Kutta integrators RK4 and RK4S. RK4 is a naive fixed step size integrator, and RK4S has automatic step size control. For two simple example problems, for which we know the solutions, the fourth order Runge Kutta integrators gave surprisingly poor results. We also studied a chaotic system described by the Lorenz model, and the results by RK4 and RK4S show large deviations from the result by RK from a surprisingly early time.

For beam physics computations, there are many cases when we require high accuracy in numerical integration. There are various sources of inaccuracy for such computations. For example, we have to worry about the field precision, but widely spread fourth order Runge Kutta numerical integrators would produce much larger numerical inaccuracy.

The code of the COSY 8th order Runge Kutta integrator RK is not overly complicated even when compared to the fourth order Runge Kutta integrator RK4S that has an adaptive step size control. The CPU time consumption for RK and RK4S shows almost no difference. Thus it seems advantageous for anybody using fourth order Runge Kutta integrators to use the COSY 8th order integrator RK instead.

A Arguments of the Subroutine RK

The Fortran coded RK is written to be compatible to the COSY language written RK program supplied in the file cosy.fox in the system of COSY Infinity [2]. Users have to supply the ordinary differential equations through a subroutine, and several arguments have to be passed to the main subroutine of the integrator, RK.

```
SUBROUTINE RK(N,X0,X1,Y0, H0,HS,H1, EPS,BS,  Z,Y1,ERREST)
```

The input arguments are the following. N is the dimensionality, i.e. the number of ordinary differential equations. X0, X1 are numbers for the starting and ending times. Y0 is an array for the values of initial condition. H0, HS, H1 are used for the step size control. HS is the suggested starting step size, and H0 and H1 are the minimum and maximum allowed step sizes. EPS and BS are used for the error control. EPS is the desired local error size and should be larger than 10^{-12}, and BS is the backstep bounds and should be larger than EPS. For higher accuracy requirements, it is necessary to port the integrator to a quadruple precision environment.

The output arguments are the solution array Y1 and the estimated accumulated error ERREST. Z is an array for the internal use, and it should have the size of at least N$\times$16.

For the example computations, we used ESP=10^{-10}, and BS=20×10^{-10}. The step sizes are set as follows.

	HS	H0	H1
1D integral	0.01	0.001	0.1
Volterra eqs.	0.1	0.02	0.5
Lorenz system	0.02	0.003	0.3

References

[1] Ingolf Kübler. Master's thesis, Justus Liebig Universität Gießen, 6300 Gießen, West Germany, 1987.

[2] M. Berz. COSY INFINITY Version 8.1 - user's guide and reference manual. Technical Report MSUCL-1195, National Superconducting Cyclotron Laboratory, Michigan State University, East Lansing, MI 48824, 2001. see also http://cosy.nscl.msu.edu.

[3] M. Abramowitz and I. A. Stegun, editors. *Handbook of Mathematical Functions.* Dover, New York, 1965, 1972. Originally published by National Bureau of Standards in 1964.

[4] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes.* Cambridge University Press, Cambridge, MA, 1989.