Numerical methods for ODEs

I. Hawke, S. Husa, B. Szilágyi

September 2004

Contents

1 Numerical methods for ODEs **2**
1.1 Basic concepts 2
1.2 Convergence tests 6
1.3 Second order Runge-Kutta methods 7
1.4 Fourth order Runge-Kutta 7

1 Basic concepts

As it may be well known to the reader, "ordinary" differential equations, as ordinary as they are, may or may not be easy to solve. One reason for this is that the solution space of all possible ODE's one could think about is much richer than the set of analytic functions one can write down by combining polynomials, trigonometric functions, and whatever else. In other terms, we are better at writing down equations than at finding exact solutions for them. Which is where approximate ODE solving methods, such as numerical methods, gain a significant role.

Given the differential equation

$$y' = y^2 + 1 \tag{1}$$

(where $y' = y'(x) = \frac{d}{dx}y(x)$) what do we mean by solution? Intuitively we mean a curve (x, y) that, at each point, has the slope $\frac{dy}{dx} = y^2 + 1$. This is a local, point-wise property. One should also note that, for the ODE given by Eq. (1) at any point (x_0, y_0) one can compute the slope of the solution, and thus one can associate a solution to all such points. In order to find a particular solution (rather than a continuous set of solutions) one would have to fix the point (x_0, y_0) and then construct, by some means, the curve representing the solution through the particular point.

Say we choose

$$x_0 = 0, \quad y_0 = 0\,. \tag{2}$$

The exact solution of Eqs. (1)-(2) is

$$y = \tan(x) \tag{3}$$

We can, therefore, compare the exact answer with the answer given by various approximative methods.

The first, most obvious approximation that comes to mind is to use the ODE and construct a high-order Taylor expansion of the function $y(x)$. Namely, we can write

$$\begin{aligned} y'' &= 2yy' = 2y(y^2+1) && (4)\\ y''' &= 2y'(y^2+1) + 4y'y^2 = 2(y^2+1)(3y^2+1) && (5)\\ &\cdots \end{aligned}$$

and then write

$$y(x) = y(x_0) + y'(x_0)(x-x_0) + \frac{1}{2}y''(x_0)(x-x_0)^2 + \frac{1}{6}y'''(x_0)(x-x_0)^3 + \cdots \tag{6}$$

or, for our particular case

$$y(x) = x + \frac{1}{3}x^3 + \cdots\,. \tag{7}$$

While Eq. (7) is a good approximation of the function $\tan(x)$ around some neighborhood of $x = 0$, it is by no means a valid solution of Eq. (1). (See Fig. 1.)

Figure 1: Comparison of $y = \tan(x)$ and its Taylor series expansion

The strength and the weakness of this first, naive approach suggests an immediate improvement: rather than using a single polynomial to approximate the function $y(x)$, we could divide the domain of interest $x_0 \leq x \leq x_{final}$ up to N intervals, and use the Taylor series expansion on each of those five intervals. For sake of clearness let's set $N = 5$ and say we are interested in the domain $x = 0 \dots 1.5$. We also use this opportunity to introduce the notion of *grid-step*

$$\Delta x = (x_{final} - x_0)/N \tag{8}$$

which, for our particular case, is $\Delta x = 0.3$. The grid-step defines a grid

$$x_i = x_0 + i\,\Delta x, \quad i = 0, \cdots N\,. \tag{9}$$

Also, for sake of ease, we stop at the first term of the Taylor series expansion Eq. (7). Our new numerical algorithm proceeds as follows:

- start from the point $x_0 = 0, y_0 = y(x_0) = 0$
- compute $y'_0 = 1$
- write $y_1 = y_0 + (\Delta x)y'_0$
- compute $y'_1 = y_1^2 + 1$
- write $y_2 = y_1 + (\Delta x)y'_1$
- etc.

In an iterative form we write

$$y_{i+1} = y_i + (\Delta x) \cdot y'_i\,, \quad i \geq 0 \tag{10}$$

Figure 2: Convergence of the Euler method to the exact solution $y = \tan(x)$.

This algorithm is the *Euler*-method. The accuracy of the algorithm is 1^{st} order (i.e., for small enough grid-steps, the error is proportional to $(\Delta x)^1$). An illustration of the convergence of this method can be seen in Fig. 2.

A simple and apparently advisable improvement to this algorithm would be to use the centered, second order accurate finite-difference expression

$$y_i' = \frac{y_{i+1} - y_{i-1}}{2\,(\Delta x)} + O(\Delta^2)\,. \tag{11}$$

In order to make use of Eq. (11) one needs *two* starting points, i.e., the values y_0 and y_1. The latter of these can easily be constructed by use of a Taylor expansion around $x = x_0$. Then the algorithm, in an iterative form, is

$$y_{i+1} = y_{i-1} + 2(\Delta x) \cdot y_i'\,, \quad i \geq 1 \tag{12}$$

We will refer to Eq. (12) as the *centered* method.

To compare the centered and the Euler methods we apply both of them to solve the ODE

$$y' = -y \tag{13}$$

with the initial condition

$$y(0) = 1\,. \tag{14}$$

The exact solution of Eqs. (13)-(14) is

$$y = \exp(-x) \tag{15}$$

The Euler method Eq. (10) translates into

$$y_{i+1} = y_i - (\Delta x) y_i = (1 - (\Delta x)) \cdot y_i\,, \quad i \geq 0 \tag{16}$$

while the centered method Eq. (12) becomes

$$y_1 = 1 - (\Delta x) + \frac{1}{2}(\Delta x)^2 - \frac{1}{6}(\Delta x)^3 \tag{17}$$

$$y_{i+1} = y_{i-1} - 2(\Delta x) \cdot y_i \,, \qquad i \geq 1 \,. \tag{18}$$

The algorithms were applied for the domain $0 \leq x \leq 5$, with $N = 20$. As apparent from figure 3, the higher order accuracy of the centered method implies, by no means, that it is better. What we see is a phenomena called *instability*. For a quick understanding of the problem we can take a closer look at the finite difference algorithms applied. The Euler method Eq. (16) boils down to

$$y_{i+1} = \rho y_i = \rho^2 y_{i-1} = \cdots = \rho^{i+1} \cdot y_0 \tag{19}$$

where $\rho = 1 - (\Delta x) < 1$. In other terms this method, applied to our particular equation, amounts to constructing a geometrical series with a basis $0 < \rho < 1$. Thus the monotonically decaying behavior is expected.

In order to understand the out-of-control behavior of the centered algorithm we attempt to put Eq. (18) into the form

$$y_{i+1} + C\, y_i = \frac{1}{C}\Big(y_i + C\, y_{i-1}\Big) = y_{i-1} + \frac{1}{C}\, y_i \tag{20}$$

In order for Eq. (20) to be identical with Eq. (18) the coefficients of y_{i-1} from the two equations must be identical. Namely we must have

$$\frac{1}{C} - C = -2(\Delta x) \tag{21}$$

The solutions of Eq. (21) are

$$C_\pm = (\Delta x) \pm \sqrt{1 + (\Delta x)^2} \tag{22}$$

The centered algorithm amounts to building a linear combination of two geometrical series

$$y_{i+1} + C_\pm\, y_i = \left(\frac{1}{C_\pm}\right)^i (y_1 + C_\pm\, y_0) \tag{23}$$

To understand the behavior of these series let's consider the case $\Delta x = 0.4$ (which is represented in figure 3). We get $C_\pm = \{1.48, -0.68\}$, with the corresponding series

$$y_{i+1} + 1.48\, y_i = \left(\frac{1}{1.48}\right)^i (y_1 + 1.48\, y_0) \tag{24}$$

$$y_{i+1} - 0.68\, y_i = \left(\frac{-1}{0.68}\right)^i (y_1 - 0.68\, y_0) \tag{25}$$

The statement is that as long as we use the centered algorithm, at each iteration the quantity $y_{i+1} + 1.48\, y_i$ will decrease by a factor of 1.48 (which is quite OK), but the quantity $y_{i+1} - 0.68\, y_i$ will flip sign at each iteration and its magnitude will increase by a factor of $1/0.68 \approx 1.47$ (which is quite BAD). Note the negative coefficient present in the bad mode – this explains why the bad quantity is not

Figure 3: Stability of the centered algorithm, illustrated for the ODE $y' = -y$.

y_{i+1} itself but rather it's weighted difference from the previous value y_i. The result is a flip-flop mode obvious in figure 3.

Theoretically one can select or suppress the OK or BAD modes by appropriate choice of initial data. In other terms, setting

$$y_1 = -C_- \, y_0 \approx 0.68 \, y_0 \tag{26}$$

means that the BAD quantity is initialized to zero and *stays* zero. However, even if the initial data was chosen with great care, the bad mode will always be present (or excited) at least at the machine roundoff level.

For all these reasons the centered algorithm, as presented above, is unusable.

The example we have looked at is an illustration of the non-trivial nature of one choosing appropriate numerical schemes. Obviously, an integration algorithm being good or bad depends not only on the approximation scheme itself but also on the ODE the scheme is being applied to.

In our case a direct analysis of the iterative algorithms was enough for understanding the stability of the systems we looked at. This does not always work easily, and there are generic theorems and methods, that can be used for this kind of analysis. As a very brief list of references one can use [5] (which was the primary source in writing up this section) as well as [6] and the references therein.

1.1 Convergence tests

There are many ways to test a numerical code for validity and accuracy, one of the most standard and most important tests is a convergence test. The basic idea is to run a code with different resolutions, and check that the convergence rate is consistent with the theoretical prediction. This serves as a test that the

algorithm has been implemented correctly, data are chosen in the convergent regime, and yields an estimate of the numerical error.

As an example for how to perform such tests, assume a quantity x that is second-order accurate: $x(t) = x_0(t) + E(t)(\Delta t^2) + O(\Delta t^3)$, x_0 denotes the exact solution, which in practice often is unknown, E denotes the unknown error term. If x_0 is known, it is sufficient to compare just 2 resolutions, x_1 computed using Δt, and x_2 computed using $\Delta t/2$. Then in the convergent regime theory predicts

$$\frac{x_1 - x_0}{x_2 - x_0} = \frac{\Delta t^2 + O(\Delta t^3)}{\Delta t^2 + O(\Delta t^3)} = \frac{\Delta t^2 + O(\Delta t^3)}{\frac{1}{4}\Delta t^2 + O(\Delta t^3)} = 4 + O(\Delta t^3).$$

When the exact solution is unknown, we can do a Cauchy-type convergence test, using at least 3 different resolutions. We use x_1 and x_2 as above, and furthermore x_4 which has been computed using $\Delta t/4$

$$\frac{x_1 - x_2}{x_2 - x_4} = \frac{\Delta t^2 - \frac{1}{4}\Delta t^2 + O(\Delta t^3)}{\frac{1}{4}\Delta t^2 - \frac{1}{16}\Delta t^2 + O(\Delta t^3)} = \frac{\Delta t^2 + O(\Delta t^3)}{\frac{1}{4}\Delta t^2 + O(\Delta t^3)} = 4 + O(\Delta t^3).$$

Note that you have to be careful at which points t you compare the solution – regardless of your choice of Δt, you always have to compare $x(t)$ at the same values of t!

Typical ways of performing the convergence test are (i) compute the convergence factor, (ii) overlay the differences of the error, which the difference of the higher resolutions multiplied appropriately.

1.2 Second order Runge-Kutta methods

We list here two different ways to implement the RK2 scheme, which you can try and implement (if you have time or later after the school) and check the claim that these methods are unstable for our model problem. In the notation of equation (29) the methods are

$$\begin{aligned} k_1 &= S(t^{n-1}, f^{n-1}) \\ k_2 &= S\left(t^{n-1} + \frac{\Delta t}{2}, f^{n-1} + \frac{\Delta t}{2} k_1\right) \\ f^n &= f^{n-1} + \Delta t\, k_2 + O(\Delta t^3) \end{aligned} \tag{27}$$

and

$$\begin{aligned} k_1 &= S(t^{n-1}, f^{n-1}) \\ k_2 &= S(t^{n-1} + \Delta t, f^{n-1} + \Delta t\, k_1) \\ f^n &= f^{n-1} + \frac{\Delta t}{2}(k_1 + k_2) + O(\Delta t^3). \end{aligned} \tag{28}$$

1.3 Fourth order Runge-Kutta

There are many different ways of organizing an RK4 ODE integration scheme. In this exercise we will implement the classical RK4 formula to integrate the

state vector f^{n-1} one timestep to obtain f^n:

$$\begin{aligned}
k_1 &= S(t^{n-1}, f^{n-1}), \\
k_2 &= S\left(t^{n-1} + \frac{\Delta t}{2}, f^{n-1} + \frac{\Delta t}{2} k_1\right), \\
k_3 &= S\left(t^{n-1} + \frac{\Delta t}{2}, f^{n-1} + \frac{\Delta t}{2} k_2\right), \\
k_4 &= S\left(t^{n-1} + \Delta t, f^{n-1} + \Delta t\, k_3\right), \\
f^n &= f^{n-1} + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) + O(\Delta t^5). \qquad (29)
\end{aligned}$$

Acknowledgements

We thank Erik Schnetter for helpful comments.

References

[1] R. C. Tolman, Phys. Rev. **55**, 364 (1939).

[2] J. R. Oppenheimer and G. Volkoff, Phys. Rev. **55**, 374 (1939).

[3] R. M. Wald. General Relativity, University of Chicago Press, Chicago, 1984.

[4] J. A. Font, M. Miller, W. Suen and M. Tobias, Phys. Rev. **D61**, 044011 (2000).

[5] R. W. Hamming. Numerical Methods for Scientists and Engineers, Second Edition, Dover Publications, New York, 1973.

[6] W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling. Numerical Recipes, Cambridge University Press, Cambridge, England, 1986.

[7] J. A. Font, T. Goodale, S. Iyer, M. Miller, L. Rezzolla, E. Seidel, N. Stergioulas, W. Suen and M. Tobias, Phys. Rev. **D65**, 084024 (2002).