ADVANCES IN ATMOSPHERIC SCIENCES, VOL. 23, NO. 5, 2006, 758–766

Analysis and Application of Multiple-Precision Computation and Round-off Error for Nonlinear Dynamical Systems

WANG Pengfei*[1] (王鹏飞), HUANG Gang[1] (黄 刚), and WANG Zaizhi[2] (王在志)

[1] *State Key Laboratory of Numerical Modeling for Atmospheric Sciences and Geophysical Fluid Dynamics* (*LASG*), *Institute of Atmospheric Physics, Chinese Academy of Sciences, Beijing* 100029

[2] *National Climate Center, Beijing* 100081

(Received 8 October 2005; revised 25 April 2006)

ABSTRACT

This research reveals the dependency of floating point computation in nonlinear dynamical systems on machine precision and step-size by applying a multiple-precision approach in the Lorenz nonlinear equations. The paper also demonstrates the procedures for obtaining a real numerical solution in the Lorenz system with long-time integration and a new multiple-precision-based approach used to identify the maximum effective computation time (MECT) and optimal step-size (OS). In addition, the authors introduce how to analyze round-off error in a long-time integration in some typical cases of nonlinear systems and present its approximate estimate expression.

Key words: multiple-precision numerical calculation, round-off error, nonlinear dynamical system

doi: 10.1007/s00376-006-0758-y

1. Introduction

Qualitative approaches are adopted in nonlinear differential equation studies. According to differential equation theory, such equations have a unique continuous solution when the Lipschitz condition is fulfilled. Numerical calculation is another popular approach, but since round-off error exists in the floating point calculations, the result is simultaneously influenced by both the discretization error and the round-off error. Studies have been made on the error analysis of numerical methods in Ordinary Differential Equations (ODEs), and the classical results of discretization error can be found in Henrici (1962), Henrici (1963), and Gear (1971). Henrici (1962, 1963) also studied round-off error on a fixed-point machine using probability theory, but the influence of round-off error on long-time numerical integrations was not investigated. Faye et al. (1985) introduced the Permutation-Perturbation method which is very efficient for evaluating round-off error and consequently for estimating the exact significant decimal figures of the algorithm's result. Vignes (1988, 1993) introduced tools for automatic implementation of stochastic arithmetic and for evaluating the accuracy of results provided by direct algorithms performed by a computer. These tools can also determine the optimal step or the optimal mesh in the approximate algorithms. Neto and Rao (1990) proposed a stochastic approach to estimate the global errors, especially in the integration situations that are often met in flight mechanics and control problems. This procedure models the errors through the distribution of zero-mean random variables belonging to stochastic sequences. The Rigorous Error Analysis method is a new technology in numerical algorithms, and complicated dynamical systems can be rigorously analyzed by means of Conley index theory (Mrozek 1996; Berger, 1999).

This research intends to reduce round-off error in nonlinear dynamical systems by applying the multiple-precision approach based on Li et al.'s experiments (2000). The latter's experiments showed that single-and-double-precision floating point operations have important effects on the long-time numerical integration in nonlinear systems. We improve their experiments as follows. In addition to both single and double precisions, multiple precisions are also used to conduct further analysis for Eq. (1) given below and to establish the relationship between numerical results using different precisions and step-sizes. We not only evaluate the round-off error propagation but also propose a way to compute the reliable numerical solution of

*E-mail: wpf@mail.iap.ac.cn

nonlinear dynamical systems over a long time.

The Optimal Step-size (OS, where in this paper, the step-size means time step-size unless stated otherwise) and Maximum Effective Computation Time (MECT) searching scheme is important for the computation of nonlinear dynamical systems. Li et al. (2000) identified OS and MECT by an optimal searching method. Further, Li et al. (2001) obtained the formulas of OS and MECT through theoretical analysis. In this paper, we will demonstrate a new multiple-precision-based approach and therefore provide more alternatives for identifying OS and MECT.

2. Multiple-precision experiments with a nonlinear system

A computer usually supports both single- and double-precision floating-point operations defined by the IEEE Standard 754[a], and some computers also support quadruple precision. Special approaches are needed for higher precision. In the computer sciences, there are special libraries to support multiple-precision computations (Oyanarte, 1990), such as GMP (GNU[b] multiple-precision arithmetic library)[c], apfloat (High Performance Arbitrary Precision Arithmetic Package for C++ and Java)[d], MAPM (My Arbitrary Precision Math library, and MPFR (The Multiple Precision Floating-Point Reliable Library)[e]. Though these libraries are different, all of them can do computing in a defined precision. Some other application software, such as Matlab[f] and Maple[g] can also carry out multiple-precision computations.

We applied the MPFR/GMP library to these experiments. The MPFR library is a C library for multiple-precision floating-point computations with exact rounding. It is based on the GMP multiple-precision library. The main purpose of MPFR is to "provide a library for multiple-precision floating-point computation which is both efficient and has a well-defined semantics."[h] The precision in MPFR is different from the significant digits. For example, single precisions and double precisions correspond to 7.22 and 15.95 significant digits respectively, while they correspond to 24- and 53-bits precision respectively in MPFR.

The classical Lorenz equations introduced by Lorenz (1963) are given as follows:

$$\begin{cases} \dfrac{dx}{dt} = -\sigma x + \sigma y \\ \dfrac{dy}{dt} = \gamma x - y - xz \\ \dfrac{dz}{dt} = xy - \beta z \end{cases} \tag{1}$$

where σ, γ, β are nondimensional constants, and t is nondimensional time.

In this research, we first consider Eq. (1) without chaos, when $\gamma = 22.0$, $\sigma = 10.0$, and $\beta = 8/3$. The initial value is set as (5, 5, 10). Previous studies have shown that this is a bad initial value (Li et al., 2000). From the theoretical analysis (Lorenz, 1963; Sparrow, 1982), we know that Eq. (1) with the above initial value has two attractors ($x = \pm 2\sqrt{14}$).

In this paper, considering the initial value issue of Eq. (1) without chaos and with a certain precision and step-size, we define the convergence computation result as the final value when t continues to increase.

Ideally, the strict criterion for deciding the final values is that the solutions to Eq. (1) should be close to constant even when t increases. The more commonly used method is to choose a long enough t and the value at that specific t is identified as the final value. The final value in our study is the numerical result when $t \geqslant 500.0$ (t also depends on the precision and a long enough time for convergence in different experiments).

Equation (1) is a set of nonlinear differential equations, and it has no analytic solution in general. Hence we can only use numerical methods to derive the approximate solutions. In order to discuss the solution based on a multiple-precision library, a 4-th order Runge-Kutta (RK, see Press et al., 1992) method is applied as an example to perform the experiments. Since we choose the parameter without chaos, when the equation is integrated over a long time, the solution converges to an attractor.

Tables 1, 2, and 3 show numerical results obtained from three different systems: Linux, IRIX and AIX, where h is the step-size. Table 1 presents the final values of variable x with the C language and with double precision. When the step-size is 0.01, the same result is obtained in the three systems. When the step-size is 0.0099999, the values of IRIX and AIX are the same, but they are different from that obtained by Linux.

[a] http://babbage.cs.qc.edu/courses/cs341/IEEE-754references.html
[b] http://www.gnu.org/
[c] http://www.swox.com/gmp/
[d] http://www.apfloat.org/
[e] http://www.mpfr.org/
[f] http://www.mathworks.com/
[g] http://www.maplesoft.com/
[h] http://www.mpfr.org/

Table 1. Final values of variable x with the C language and with double precision.

	$h = 0.01$	$h = 0.0099999$
Linux	−7.483314773547912	−7.483314773547889
IRIX	−7.483314773547912	7.483237136577893
AIX	−7.483314773547912	7.483237136577893

Table 2. Final values of variable x with the Fortran language and with single precision.

	$h = 0.01$	$h = 0.0099999$
Linux	−7.483299	−7.483300
IRIX	7.483307	−7.483299
AIX	−7.483299	−7.483299

Table 3. Final value of variable x with the Fortran language and with double precision.

	$h = 0.01$	$h = 0.0099999$
Linux	−7.483314773547912	−7.483314773547928
IRIX	−7.483314773547912	−7.483314773547913
AIX	−7.483314773547910	7.483314773547833

Table 2 shows the final values of variable x with the Fortran language and with single precision. When the step-size is 0.01, the final values of Linux and AIX are the same, but they differ from that using IRIX. When the step-size is 0.0099999, all final values of the three systems are close. Table 3 gives the final values of variable x with the Fortran language and with double precision. When the step-size is 0.0099999, all final values of the three systems are close. When the step-size is 0.0099999, the final values of IRIX and Linux are the same, but they differ from the result of AIX.

We can see from Tables 1, 2, and 3 that the same program can lead to different results because of the floating point architecture and compiler's distinctions in the different systems. For the Lorenz equations with the initial condition above, we cannot derive the true final value with single or double precisions.

Figure 1 shows the numerical final values versus the precisions*.

When the step-size is 0.01 and the computation precision reaches a certain point, which we call $\mathrm{PREC_h}$, the final values will not bounce and will be close to the attractors. That is how we get a final value. The $\mathrm{PREC_h}$ varies with step-size h. Both $\mathrm{PREC_h}$ values in the two experiments (Figs. 1a, b) are greater than 53, the bit precision of the double precision, so only using single or double precision may not be enough to analyze these nonlinear equations.

The higher precision experiments show that the bad initial value cannot become a good initial value when the precision increases.

The solution difference with certain precision between the numerical solution and the reference value in the studies applying Lorenz equations is defined as:

$$D_{xyz} = \sqrt{\sum_{x,y,z} (V_t - V_{t0})^2} \,. \tag{2}$$

In the equation, V is variable, V_t is the numerical solution at time t, and V_{t0} is the reference value (or reference solution) at time t, which is very close to the real solution. Since we cannot get the real solution values of the equations, we will use the reference values as a substitute for the real values in the following numerical experiments.

The norm of the reference solution is:

$$A_{xyz} = \sqrt{\sum_{x,y,z} (V_{t0})^2} \,, \tag{3}$$

and then the relative error of the numerical solution becomes:

$$R = D_{xyz}/A_{xyz} \,. \tag{4}$$

Further, to keep the relative error of the numerical solution R within a limited range (such as $R < 1/10$), the effective computation time (ECT) is needed.

The ECT is related to the numerical method and step-size. Thus the MECT is the maximum time among all ECTs.

From Eq. (1) and the above parameters, the numerical solution of time t can be achieved when the precision varies. Here, we only explore the results when time t is set to 30 (Fig. 2) .

Figure 2 depicts that the two numerical solutions at $h = 0.01$ and $h = 0.0099999$ vary prominently when the precision is less than 50. On the contrary, both solutions are similar and almost without variation after 50 bits of precision. Therefore, the precision at 50 can be regarded as the effective computation precision (ECP).

However, even in the ECP, there also exits a difference between these two step-sizes (Fig. 2), suggesting that the total error is mainly affected by step-size, and the step-sizes 0.01 and 0.0099999 may not be the convergent step-sizes. In other words, we have to decrease the step-size to get the real values.

*When MPFR/GMP is applied in the computation, the results from the different systems are the same. Hence the following experiments use the result from Linux as an example. To avoid redundant output, the program uses multiple-precision in the computing process, but some outputs are presented with double precision

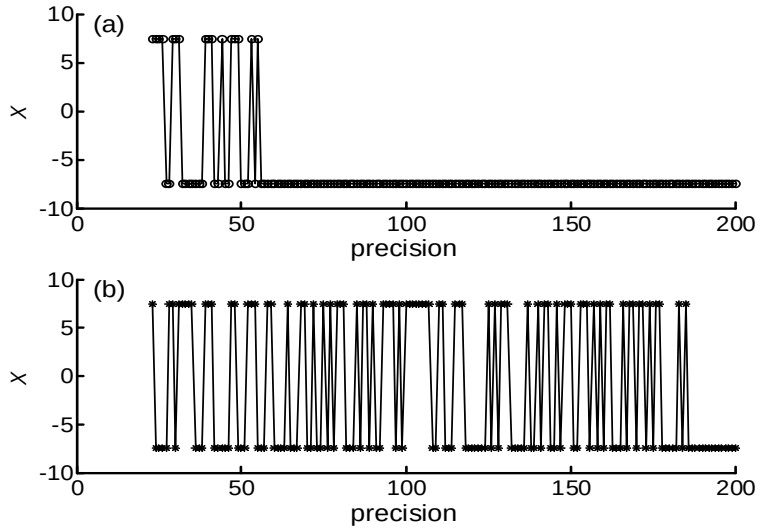

Fig. 1. The final values versus the precisions, (a) $h = 0.01$ and (b) h=0.0099999; $\gamma = 22.0$.

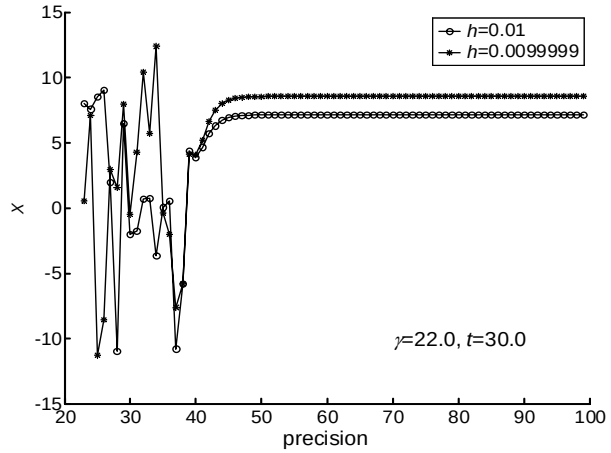

Fig. 2. The numerical solution of time t as the precision changes, $h = 0.01$ and $h = 0.0099999, t = 30.0$.

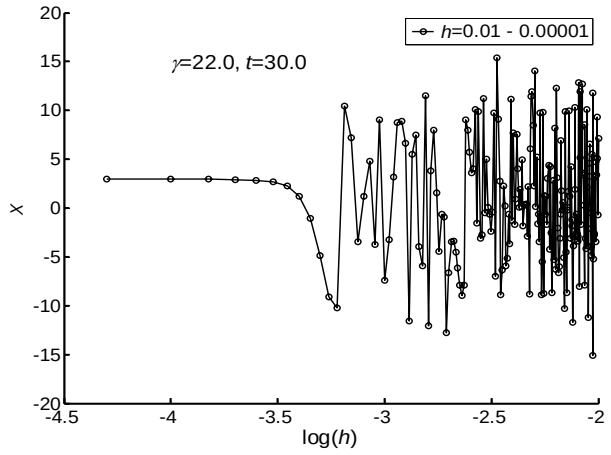

Fig. 3. The numerical solution of time t as the step-size h changes, $t = 30.0$, precision=100.

Given the time ($t = 30$), precision=100 (within ECP), the numerical solution varies as the step-size ranges from ($h = 0.01$) to ($h = 0.0001$). Note that the numerical solution does not converge when h is not small enough ($h > 0.0002$), and it changes with no evident pattern. When the step-size h is less than 0.0002, the numerical solutions tend be constant.

Similar results have also been obtained when the same experiments are repeated with other initial values and step-sizes. The experiment results of Eq. (1) without chaos and with a bad initial value are as follows:

(1) The ECT exists in the numerical solution of the nonlinear equation. With a certain computing precision, the numerical result is indeterminate and sensitive to step-size when t is beyond ECT.

(2) Different ECP values exist in the numerical solution of the nonlinear equation corresponding to different time t values. We can get the numerical solution close to the real value only if the precision we use in the method is equal to or greater than the ECP.

(3) The numerical result is determinate when the initial value, method, and step-size are given while the precision is infinite, but the result may not be the real value.

(4) A suitable step-size is needed to get the correct numerical solution when both the ECT and ECP are satisfied. However, not all of the step-sizes can lead to the right result, except when the step-size is close to the OS.

There are many similarities in the experiments of the chaotic case within ECT. The main difference between the chaotic case and non-chaotic case is that the numerical solutions do not convergent to a final

value, and they change continuously without any distinct pattern.

All the experiments with chaos and without chaos show that a numerical solution with an accuracy of a certain number of digits is determinate when the initial value, method, and step-size are given and the precision is infinite. Furthermore, a numerical solution with a given accuracy is determinate when the initial value, method and step-size are certain and the precision is big enough. So we can use a finite precision to do research when we only need a solution of a given number of digits of accuracy.

The steps to compute the real numerical solution of Lorenz systems over a long time are: First, choose a normal step-size (such as 0.01), increase the computing precision, and then obtain the value of the numerical solution with a precision so as to obtain the ECP. Second, keep the precision higher than the ECP and decrease the step-size to get the value of the numerical solution with step-size. The convergent step-size should then be found (called the ECS). Lastly, compute with a precision higher than the ECP and a step-size less than the ECS to get a real numerical solution. How close it is to the real value depends on the precision and step-size used.

3. Error analysis of some nonlinear systems

Considering the long-time error of nonlinear Eq. (1), we can use the following formula to describe the total error:

$$E = e(t,h) + e(r,t)\,, \tag{5}$$

where $e(t,h)$ is the discretization error for nonlinear ODEs,

$$\|e(t,h)\| \leqslant N_1(\eta) e_{(0)} e^{kL\Gamma^*(t-t_0)} + Ch^p \frac{e^{kL\Gamma^*(t-t_0)}-1}{kL}\,, \tag{6}$$

where η is variable; $N_1(\eta) = 1$ when $\eta \leqslant 1$, and $N_1(\eta) = \eta$ if $\eta > 1$; k means k step method; $e_{(0)} = \max\limits_{0<j\leqslant k-1} \|e_j\|$ such that e_j is the maximum initial error of the k step method; L is the Lipschitz constant;

$$\Gamma^* = \frac{\Gamma}{1 - h\beta|\alpha_k|^{-1}}\,;$$

h is step-size; and β are α_k are constants; $\Gamma = \sup\limits_{j=0,1,2,\ldots} |r_j| < \infty$; r_j are coefficients; C is constant; and p is the order of the method. More details can be found in Li et al. (2001).

For a single-step method such as the 4-th order Runge-Kutta method, $e_{(0)} = 0$, and the above formula can be simplified as:

$$\|e(t,h)\| \leqslant Ch^p \frac{e^{kL\Gamma^*(t-t_0)}-1}{kL}\,. \tag{7}$$

$e(t,h)$ can be understood as:

$$e(t,h) = Ch^p f(t) + \varepsilon\,. \tag{8}$$

In Eq. (8), p is the order of the method,

$$f(t) = \frac{e^{kL\Gamma^*(t-t_0)}-1}{kL}\,,$$

and ε is a small quantum compared to the first item.

$e(r,t)$ is round-off error, and it corresponds to the precision and time, and r is the precision in bits.

Considering the real value of t, we can write $x(t)$ as x, and $X(t,h,r)$ is the numerical solution. We will further discuss the application of Eqs. (7) and (8) in the following three typical cases. The discussion is within ECT.

Typical Case 1:

How can we obtain a better solution when h is close to OS with the same time and precision r but different step-size?

$$\begin{cases} X_1 - x = Ch_1^p f(t) + \varepsilon + e(r,t)\,, \\ X_2 - x = Ch_2^p f(t) + \varepsilon + e(r,t)\,, \end{cases} \tag{9}$$

where $X_1 = X(t,h_1,r)$, $X_2 = X(t,h_2,r)$, $x = x(t)$.

When the precision r is very high, and the $e(r,t)$ becomes a tiny value compared to $e(t,h)$, we get the approximate equation (we can obtain the relationship expressed below):

$$\begin{cases} X_1 - x = Ch_1^p f(t) + \varepsilon\,, \\ X_2 - x = Ch_2^p f(t) + \varepsilon\,. \end{cases} \tag{10}$$

We can get a series of X_i by changing h_i. No matter whether the order p of the numerical method is known or not, we can use a curve-fitting method to obtain the approximate numerical solutions.

The above discussion can be used to get an approximate numerical solution close to the real value faster. In order to get the best result in the multiple-precision method, we need to choose a very small step-size (close to OS), which takes the CPU much time to finish the computation. The experiments show that when the step-size is small enough, the results of the various step-sizes are close to the real values. Hence the results obtained by using these stepsizes can lead to a better solution, and save more CPU time.

Typical Case 2:

Consider the equation within ECT when the step-size h and precision r are the same, but the time t is different:

$$\begin{cases} X_1 - x(t_1) = Ch^p f(t_1) + \varepsilon + e(r,t_1)\,, \\ X_2 - x(t_2) = Ch^p f(t_2) + \varepsilon + e(r,t_2)\,. \end{cases} \tag{11}$$

In Eq. (11), $X_1 = X(t_1,h,r)$, $X_2 = X(t_2,h,r)$, $x = x(t)$.

Table 4. The numerical solutions vary with the precision, $h = 0.0001$.

Precision (bits)	t	h	X
100	30.0	0.0001	2.9363712434500740
101	30.0	0.0001	2.9363712434500732
102	30.0	0.0001	2.9363712434500728
103	30.0	0.0001	2.9363712434500726
104	30.0	0.0001	2.9363712434500725
105	30.0	0.0001	2.9363712434500724
106	30.0	0.0001	2.9363712434500724
107	30.0	0.0001	2.9363712434500724
108	30.0	0.0001	2.9363712434500724
109	30.0	0.0001	2.9363712434500724

Table 5. The numerical solutions vary with the precision, $h = 0.00001$.

Precision (bits)	t	h	X
100	30.0	0.00001	2.9378871138255342
101	30.0	0.00001	2.9378871138255261
102	30.0	0.00001	2.9378871138255222
103	30.0	0.00001	2.9378871138255202
104	30.0	0.00001	2.9378871138255192
105	30.0	0.00001	2.9378871138255187
106	30.0	0.00001	2.9378871138255184
107	30.0	0.00001	2.9378871138255183
108	30.0	0.00001	2.9378871138255182
109	30.0	0.00001	2.9378871138255182

Table 6. The numerical solutions vary with the step-size.

Precision (bits)	h	t	X
100	0.000010	30.0	2.9378871138255342
100	0.000009	30.0	2.9378871656790527
100	0.000008	30.0	2.9378872028521291
100	0.000007	30.0	2.9378872284171930
100	0.000006	30.0	2.9378872450860222
100	0.000005	30.0	2.9378872552095293
100	0.000004	30.0	2.9378872607775470
100	0.000003	30.0	2.9378872634186109
100	0.000002	30.0	2.9378872643997443
100	0.000001	30.0	2.9378872646262659

When the precision r is high, and the value of $e(r,t)$ becomes a tiny value compared to $e(t,h)$, so we can obtain the relationship expressed below:

$$\begin{cases} X_1 - x(t_1) = Ch^p f(t_1) + \varepsilon \,, \\ X_2 - x(t_2) = Ch^p f(t_2) + \varepsilon \,. \end{cases} \tag{12}$$

And we get:

$$\frac{X_1 - x(t_1)}{X_2 - x(t_2)} = \frac{Ch^p f(t_1) + \varepsilon}{Ch^p f(t_2) + \varepsilon} \,. \tag{13}$$

From the above expression, we can know that the effect of discretization error becomes larger and larger.

The discretization error versus t can be fitted from the numerical solution of X_1 with a bigger step-size and the value of X_2 with a tiny step-size which is close to the OS.

Typical Case 3:

How can we get a better solution when $r \to \infty$ with the same time and step-size but different precisions r_1 and r_2?

$$\begin{cases} X_1 - x = Ch^p f(t) + \varepsilon + e(r_1, t) \,, \\ X_2 - x = Ch^p f(t_2) + \varepsilon + e(r_2, t) \,, \end{cases} \tag{14}$$

where $X_1 = X(t, h, r_1)$, $X_2 = X(t, h, r_2)$, $x = x(t)$. It is easy to get:

$$X_1 - X_2 = e_{r1} - e_{r2} \,. \tag{15}$$

When the precision is very large, $e_r \to 0$, and we can use curve fitting to get the approximate e_r.

After we get the e_r, we rewrite the equation $X_1 - x = Ch^p f(t) + \varepsilon + e(r,t)$ in (14) as:

$$x + Ch^p f(t) + \varepsilon = X_1 - e(r,t) \,. \tag{16}$$

In Eq. (16), the left side of the equation is the numerical solution that only includes discretization error.

When the e_{r2} in Eq. (15) is very small, $X_1 - X_2$ can be considered the approximate value of e_{r1}. Changing the step-size h, while keeping $h_1 = h_2 = h$, we can get the curve of e_{r1} which corresponds to step-size h. Changing t, while keeping $t_1 = t_2 = t$, we can get the curve of e_r which corresponds to time t.

4. Application of the error analysis method

4.1 *To calculate the approximate numerical solution*

We can use the error analysis method to calculate the approximate numerical solution of a certain time with multiple precisions.

Tables 4 and 5 are the numerical results of Eq. (1), and the initial values are the same as in Table 1, $t = 30$. The precision ranges from 100 to 109 bits, and the step-size is 0.0001 and 0.00001.

Tables 4 and 5 indicate that the difference between the numerical solutions of a 100-bit precision and of a higher precision is within 10^{-14}.

In the next step, we keep the precision at 100 bits and change the step-size smoothly from 0.00001 to 0.000001; now we see the differences between numerical solutions are about 10^{-9} (Table 6), which is much greater than the difference of precision (10^{-14}) when changing from 100 bits to infinite precision. At this time, the way to decrease the total error is to use a smaller step-size. Using the approximate formula of OS from the paper by Li et al. (2001), we know that

the OS is about 10^{-3} at single precision, and we can estimate the OS at 100-bit precision is about 10^{-11}.

The CPU time used for the computation can be described as $T = t/h \times T_0$, where T_0 is the time for the program to finish one computing step. If the step-size h decreases by 10^{-5} times, the total CPU time will increase by 10^5 times.

We use the values of step-size h=0.000001, 0.000002, 0.000003,. . ., to conduct a curve-fitting and get the approximate solution.

Transform Eq. (9) to:

$$X_1 = x + Ch_1^p f(t) + \varepsilon + e(r,t)\ , \tag{17}$$

use the $p+1$ level approximate of h_1:

$$X_1 = x + Ch_1^p f(t) + C_2 h_1^{(p+1)} f(t)\ , \tag{18}$$

and when $t = 30$, $f(t)$ is constant, so set $C' = Cf(t)$ and $C' = C_2 f(t)$ such that

$$X_1 = x + C'h_1^p + C_2' h_1^{p+1}\ . \tag{19}$$

In this paper, we use the 4-th order RK method, $p = 4$. Because a high precision curve-fitting is needed, we use Maple to do the curve-fitting.

In order to check the effect of fitting more easily, we use the quadruple precision numerical solution to do the curve-fitting. Table 7 list the five values of step-size h=0.000005,. . ., 0.000001.

The fitting result using Maple with 50 significant digits* is:

$$\begin{aligned} X =& 2.9378872646412044651622292881100578732173913043481 - \\ & 15099915434656.617445520360608695652173922163896956 \times h^4 + \\ & 1837038259144639.828787391304347826091369627557854 8 \times h^5\ , \end{aligned}$$

Where 2.937887264641204465162229288110057 can be regarded as the approximate numerical solution with 113-bit precision.

4.2 *To establish the multiple-precision method to determine OS and MECT*

Li et al. (2000, 2001) identified OS and MECT by an optimal searching method. The program they used was written in Fortran under a SGI Origin 2000 computer, in which the system's single and double precisions have 7.22 and 15.95 significant digits. They found the OS to be about 0.005–0.006 and MECT to be about 17 with single precision.

The initial value and parameters in Fig. 4 are the same as Fig. 1 except $r = 28.0$. Figure 4 shows that the R values at single and double precisions are close in the early stages of the computation time (a–f), but after a certain time, the R values become different, and the result of single precision is changed to no rule earlier than for double precisions.

The experiments imply that for the same step-size h, and precisions $p_1 < p_2$, when time increases from 0 to greater values, the numerical solution at precision p_1 causes a departure from the real value earlier than precision p_2 does.

Since the result under the precision p_2 is more accurate than that of precision p_1, we can use the value where the step-size is h_2 ($h_2 \ll h$) and precision is p_2 as the reference value, and then calculate out the R at the same time. R varies from time 0, so when R is greater than 1/10, we regard the time as the ECT of precision p_1 with step-size h. When changing step-size h, such as from 0.1 to 0.000001, we get many ECTs; the greatest of these ECTs is the Maximum ECT (MECT) of precision p_1, and the relevant step-size is the OS.

Table 7. The data used for the curve-fitting.

Precision (bits)	h	t	X
113	0.000005	30.0	2.937887255209497411599097709935623 5
113	0.000004	30.0	2.937887260777507169788663975924144 1
113	0.000003	30.0	2.937887263418557715252015422501304 0
113	0.000002	30.0	2.937887264399664603432016037456253 3
113	0.000001	30.0	2.937887264626106386765831815304366 3

*We use the Least Squares function in Maple to do the curve-fitting

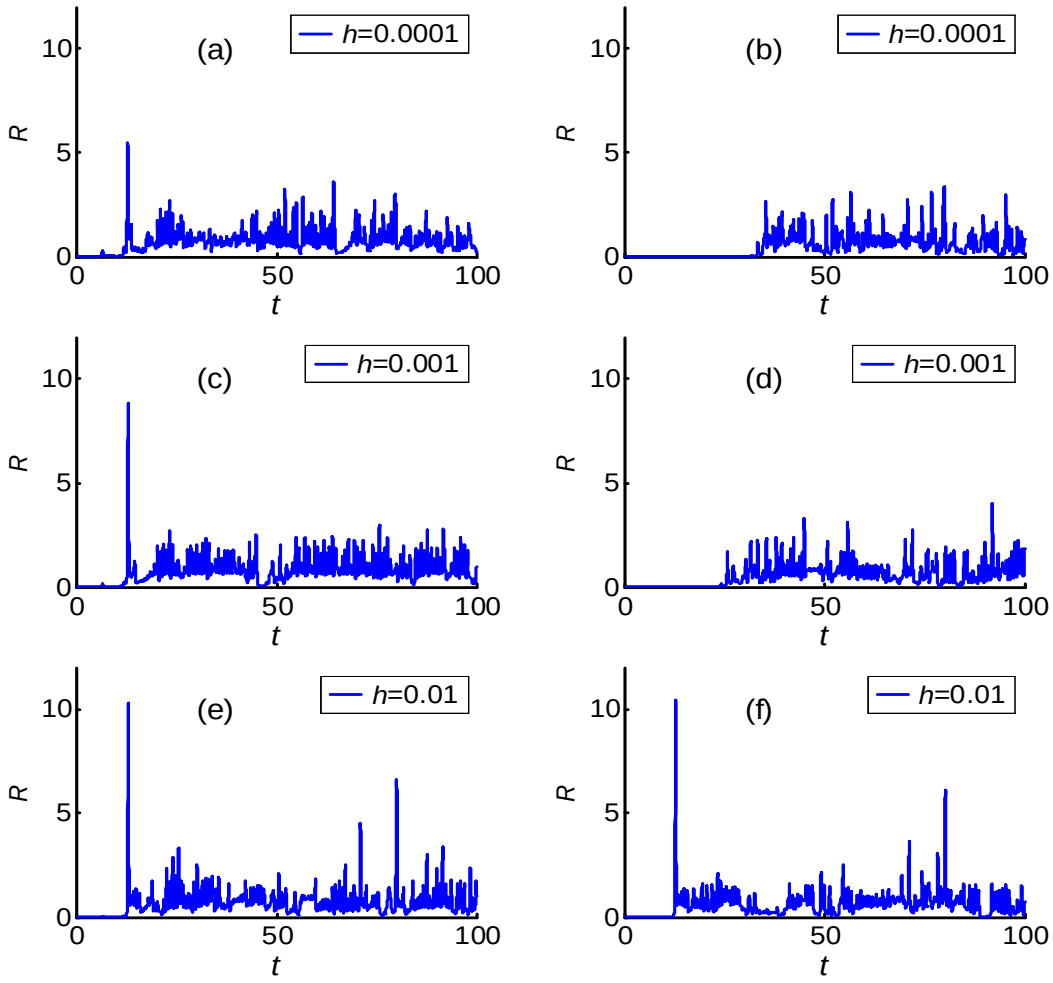

Fig. 4. The relative error R varies with respect to time, (a, c, e) is at single precision, (b, d, f) is at double precision, $r = 28.0$.

Table 8. Comparing the real computation value and the fitted value.

h	X computed	X fitted	Fitting error
0.0000001	2.937887264641202953376710391907779 6	2.937887264641202955189056205039759 7	$\sim 10^{-17}$
0.00000001	2.937887264641204477777660951516018	2.937887264641204465011230317467317 6	$\sim 10^{-16}$

5. Conclusion and discussion

It may not be enough to use single or double precision in the analysis of nonlinear equations. The multiple-precision program can analyze them with higher precision, especially when a long-time integration is involved.

The results of higher precision experiments indicate that bad initial values cannot become good initial values as the precision increases.

Moreover, the procedure for obtaining a real numerical solution of the Lorenz system with long-time integration is presented in the paper. The numerical results may be incorrect if a low precision or a step-size not chosen carefully are used to study the Lorenz system's long-time status (such as $t > 30.0$). Though we can get the MECT and OS from the theoretical formula, the computing process is very complex. The OS and MECT detection method introduced in this paper can enable the computer to identify them automatically.

The influence of round-off error in nonlinear dynamical systems should be well recognized. In this paper, three analyses of a simple question are put forward, and the approximate method can be used in some multiple-precision operations.

The computing precision affects the nonlinear equations and even the numerical model computation. Currently, most computer systems only support limited hardware precision (such as single, double and quadruple precision). It can help us perform the round-off error analysis easier and faster if multiple-precision is supported by the hardware. Thus, scalable precision computer systems and compiler software should be worked on. The multiple-precision method used in this paper is CPU-time intensive, especially when the precision is higher than 200 bits or when doing an OS search. Using a parallel computing method can help to save computation time and increase efficiency.

Acknowledgments. This study was supported by the National Key Basic Research and Development Project of China 2004CB418303, the National Natural Science foundation of China under Grant Nos. 40305012 and 40475027, and Jiangsu Key Laboratory of Meteorological Disaster KLME0601. The authors would like to thank Prof. Ronghui Huang and Prof. Jianping Li for their useful suggestions that helped improve this manuscript.

REFERENCES

Berger, A., 1999: Rigorous error bounds for RK methods in the proof of chaotic behavior. *Journal of Computational and Applied Mathematics*, **111**(1–2), 13–24.

Faye, J. P., and J. Vignes, 1985: Stochastic approach of the permutation-perturbation method for round-off error analysis. *Applied Numerical Mathematics*, **1**(4), 349–362.

Gear, C. W., 1971: *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice-Hall, Englewood Cliffs, 253pp.

Henrici, P., 1962: *Discrete Variable Methods in Ordinary Differential Equations*. John Wiley, New York, 187pp.

Henrici, P., 1963: *Error Propagation for Difference Methods*. John Wiley, New York, 73pp.

Li Jianping, Zeng Qingcun, and Chou Jifan, 2000: Computational uncertainty principle in nonlinear ordinary differential equations—I Numerical Results. *Science in China* (Series E), **43**(5), 449–461.

Li Jianping, Zeng Qingcun, and Chou Jifan, 2001: Computational uncertainty principle in nonlinear ordinary differential equations—II Theoretical analysis. *Science in China* (Series E), **44**(1), 55–74.

Lorenz, E. N., 1963: Deterministic nonperiodic flow. *J. Atmos. Sci.*, **20**, 130–141

Mrozek, M., 1996: Rigorous error analysis of numerical algorithms via symbolic computations. *Journal of Symbolic Computation*, **22**, 435–458.

Neto, A. R., and K. R. Rao, 1990: A stochastic approach to global error estimation in ODE multistep numerical integration. *Journal of Computational and Applied Mathematics*, **30**(3), 257–281.

Oyanarte, P., 1990: MP-A multiple precision package. *Computer Physics Communications*, **59**(2), 345–358.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, 1992: *Numerical Recipes in C*. Cambridge University Press, Cambridge, 965pp.

Sparrow, C., 1982: *The Lorenz Equations: Bifurcations, Chaos and Strange Attractors*. Springer-Verlag, New York, 269pp.

Vignes, J., 1988: Review on stochastic approach to round-off error analysis and its applications. *Mathematics and Computers in Simulation*, **30**(6), 481–491.

Vignes, J., 1993: A stochastic arithmetic for reliable scientific computation. *Mathematics and Computers in Simulation*, **35**(3), 233–261.