Computer Physics Communications 317 (2025) 109855

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Computational Physics

An automated parallel program of Clean Numerical Simulation for chaotic systems governed by ODEs

Bo Zhang [a], Shijun Liao [a,b,*]

[a] State Key Laboratory of Ocean Engineering, Shanghai 200240, China
[b] School of Ocean and Civil Engineering, Shanghai Jiaotong University, Shanghai 200240, China

ARTICLE INFO

The review of this paper was arranged by Prof. Peter Vincent

Dataset link: https://github.com/sjtu-liao/cnspy

Keywords:
Chaos
Python-defined ODEs
Clean Numerical Simulation

ABSTRACT

Due to the butterfly-effect, numerical noise in chaotic systems grows exponentially, presenting a significant challenge. This issue can be mitigated through the use of Clean Numerical Simulation (CNS) proposed by Liao in 2009, which can effectively reduce numerical noise to a desired (say, arbitrarily low) level in an interval of time that is long enough for statistics. In this paper, we propose the `CNSPy`, a novel, highly efficient, self-adaptive CNS implementation to obtain the convergent (i.e. reproducible) numerical simulation of chaotic systems governed by a set of ordinary differential equations (ODEs). This software automates the CNS computational workflow by automatically converting Python-defined ODEs into a parallelized C code, eliminating the need for error-prone manual derivation and coding while ensuring high efficiency in high-performance computing (HPC) environments. The code is free and available at https://github.com/sjtu-liao/cnspy.

1. Introduction

It is well established that the sensitivity on initial conditions (SDIC) is one of the most challenging characteristics of chaotic systems [1–3]. Even the smallest variation in initial conditions or system parameters can lead to drastically different outcomes [4,5]. Importantly, such variations can arise from numerical errors introduced by different computational methods or varying levels of precision, such as the use of `Float32` versus `Float64` in Python. This sensitivity is not only driven by the initial conditions but also by the numerical algorithms employed. As Lorenz [6] noted, a simple three-dimensional chaotic system can exhibit a phenomenon referred to as sensitivity dependence on numerical algorithms (SDNA). Numerical noise, an inevitable component of any computational process, grows exponentially and can match the magnitude of the 'true' physical solution, eventually. Moreover, the shadowing solution of the famous shadowing lemma may not reflect physical reality [7]. The unreliability of chaotic numerical solutions has long been a topic of debate regarding the credibility of numerical simulations of chaotic dynamical systems [8], with some researchers arguing that the numerical convergence of chaotic systems cannot be guaranteed indefinitely [9].

Beyond classical integer-order chaotic systems, comparable challenges emerge in fractional-order models, where memory effects and nonlocality compromise the reliability of long-term simulations. Recent studies demonstrate that fractional chaotic systems exhibit comparable sensitivity to numerical noise and algorithmic choices, necessitating high-accuracy numerical methods. For instance, Owolabi and Karaagac [10] investigated chaotic and spatiotemporal oscillations in fractional reaction-diffusion systems, highlighting the critical need to manage numerical noise for long-term reproducibility. Similarly, Owolabi and Atangana [11] simulated complex spatiotemporal patterns under fractional dynamics, demonstrating the essential role of high-precision numerical techniques in capturing key chaotic features. Further theoretical and numerical analyses of fractional-order systems with Riemann–Liouville derivatives [12] underscore the crucial importance of controlling round-off and truncation errors. In applied settings, such as epidemic modeling with nonlocal and nonsingular derivatives [13], maintaining numerical stability is equally imperative. Collectively, these findings underscore the critical importance of noise-sensitive and precision-focused numerical strategies for ensuring reliable long-term simulations of both integer- and fractional-order chaotic systems.

To overcome the above-mentioned difficulties, Liao [14] proposed a new numerical strategy in 2009, namely the clean numerical simulation (CNS) [15–19]. Different from other traditional numerical algorithms, CNS decreases *not only* truncation errors by means of high-order

* Corresponding author.
E-mail address: sjliao@sjtu.edu.cn (S. Liao).

https://doi.org/10.1016/j.cpc.2025.109855
Received 22 May 2025; Received in revised form 14 July 2025; Accepted 4 September 2025

Available online 16 September 2025
0010-4655/© 2025 Elsevier B.V. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

truncation numerical algorithms (such as the Taylor series method in time and the pseudo-spectral method in space) *but also* round-off error using high accurate data with a large number of significant digits. In the frame of CNS, numerical errors are controlled below such a desired level that they are negligible compared to the true solution in a *finite* interval of time $t \in [0, T_c]$, where T_c is called the critical predictable time [16]. To confirm the convergence of a chaotic numerical simulation, the same CNS algorithm with even smaller numerical noises is used to gain another simulation that should be even closer to the true solution in the same interval of time $t \in [0, T_c]$. In this way, one can gain convergent (i.e. reproducible) numerical simulations of a chaotic system in a finite interval of time that however is long enough for calculating statistics.

As mentioned in Liao's book [15], the computational efficiency of CNS has been increased several orders of magnitude since 2009 and many chaotic problems have been successfully solved by CNS, such as the Lorenz-63 system [18], the spatial-temporal chaos [19], the Arnold-Beltrami-Childress flow [20], and so on. Especially, using a strategy based on CNS, the number of planar periodic orbits of the famous three-body problem has been increased several orders of magnitude [21–23], and a roadmap to gain planar periodic orbits of general three-body system with arbitrary masses was proposed by Liao et al. [24] in 2022. Note that only thirty families of planar periodic orbits [25,26] had been identified prior to their discoveries since Newton mentioned it in 1680s. Besides, Li and Liao [27] currently discovered 10,059 *three-dimensional* periodic orbits of general three-body system by means of CNS. By means of CNS, it was found that statistic results of some chaotic systems are very sensitive to small disturbances: this fact leads to a completely new concept "ultra-chaos" [28,29] that has a higher disorder than a normal-chaos whose statistics are stable to tiny disturbances. In addition, the CNS has demonstrated significant potential in turbulence research. For example, Qin et al. [30,31] found that numerical noises as artificial stochastic disturbances have large-scale influence on a sustained turbulence. Liao and Qin discovered the so-called noise-expansion cascade of turbulence [32] that could reveal the origin of macroscopic randomness of turbulence. In 2025, Liao and Qin [33] applied CNS to provide, for the first time, a rigorous evidence that artificial numerical noise in direct numerical simulation (DNS) of turbulence is approximately equivalent to thermal fluctuation and/or stochastic environmental noise, and this provides, for the first time, a positive viewpoint regarding the presence of artificial numerical noise in DNS. Currently, Qin and Liao [34] successfully applied CNS to solve a three-dimensional turbulent flow. All of these not only validate CNS as an innovative methodology bridging computational simulations with physical reality, but also establish a conceptual framework for probing the origin of randomness in complex fluid systems.

As one can predict that a large order Taylor series method and number of significant digits in the CNS will be costly in time. Thus, some efforts have been made to accelerate the Taylor series method by many researchers. Unlike the fast Fourier transform (FFT) speed up the Fourier transform by using the symmetrical characteristic of the Fourier base, the Taylor series is hard to directly speed up the calculation. However, one can use the optimal time step and optimal Taylor series order to reduce computational effort. Scott [35] gives an optimal time step by using the (N-1)-th order Lagrange remainder of the Taylor series, and barrio [36] gives a similar optimal time step with both the N-th and (N-1)-th order. Jorba [37] gives both the optimal time step and optimal order of Taylor series method by a different estimation of the Taylor series' error. For the above optimal method for the Taylor series method, Qin [38] gives an optimal number of significant digits for CNS by considering the error growth of the objective system.

Several packages employ the Taylor series method to solve ordinary differential equations (ODEs), providing valuable tools across various domains [37,39,40]. However, these existing packages are not designed based on the CNS strategy, which is crucial for obtaining reliable solutions in chaotic systems. Notably, they typically focus on optimizing order and time step without considering the optimal number of significant digits, and the chosen optimal order may not be suitable for parallel implementations. To overcome these limitations, this paper revisits the self-adaptive CNS methodology and introduces an innovative optimization of the order parameter specifically designed for parallel computing environments. Then, we present a new implementation `CNSPy` with the new self-adaptive CNS method, and use it to solve some typical chaotic system as some examples.

Compared with conventional integrators such as Runge–Kutta schemes (e.g., RK4), CNSPy adopts a fundamentally different philosophy. Traditional solvers aim at controlling local truncation errors and are widely used for their efficiency and robustness. However, they offer no theoretical guarantee for the long-term accuracy of chaotic trajectories, as numerical noise can exponentially amplify and ultimately dominate the solution. Most mainstream ODE solvers, such as those in `SciPy`, `DifferentialEquations.jl`, or MATLAB's ODE suite, primarily support short-term simulations with adaptive step-size control. Their parallelism is often limited to ensemble runs or coarse time-domain decomposition, and they lack intrinsic support for arbitrary precision or adaptivity in digit accuracy. In contrast, `CNSPy` is specifically designed to deliver mathematically convergent solutions within a user-defined critical predictable time (T_c), by automatically estimating the required Taylor order and number of significant digits. This not only ensures numerical reliability, but also provides a natural framework for fine-grained task-level parallelism, as individual Taylor terms can be evaluated independently. This makes `CNSPy` both accurate and scalable on modern multi-core CPUs, addressing the major limitations of existing integrators when applied to long-term simulations of chaotic systems.

`CNSPy` is publicly available at https://github.com/sjtu-liao/cnspy. The repository includes detailed documentation and example scripts that reproduce all key results in this paper.

2. Review of the self-adaptive CNS

The numerical method in `CNSPy` is based on the self-adaptive CNS. The entire procedure of the self-adaptive CNS algorithm including initialization, error control, Taylor order adjustment, and step-size adaptation processes, which is illustrated in Fig. 1. So in this section, we will give a brief review of the self-adaptive CNS. For more details, please refer to Qin and Liao [38].

2.1. Basic idea of CNS

CNS is a set of strategies with the purpose of arbitrarily controlling numerical noise. Its main point is decreasing the truncation error and the round-off error to a required level. Truncation errors come from the discretization of continuous systems. Numerical methods have the following general form:

$$f(t+h) = f(t) + h \cdot RHS(t) \tag{1}$$

where $RHS(t)$ is the right-hand side, it varies according to the numerical methods. For the N-th-order Taylor series method, the right-hand side can be written as:

$$RHS(t) = \sum_{i=1}^{N} \frac{f^{(i)}(t)}{i!} \Delta t^{i-1} + \mathcal{O}(h^N) \tag{2}$$

The CNS suggests that increasing N with higher-order Taylor series methods can reduce truncation errors to a sufficiently small level. Additionally, by employing extended precision formats to store additional digits in computational systems, the number of significant digits N_s can be increased, thereby significantly reducing round-off errors. As a result, numerical noise can be controlled to an arbitrarily small magnitude.

Since numerical noise grows exponentially, there exists a critical predictable time T_c, defined as the time when the average level of numerical noise reaches a predefined threshold, known as the critical numerical noise ϵ_c. Therefore, to determine the required values of N and N_s, it is necessary to first define the relation between T_c and numerical noise.

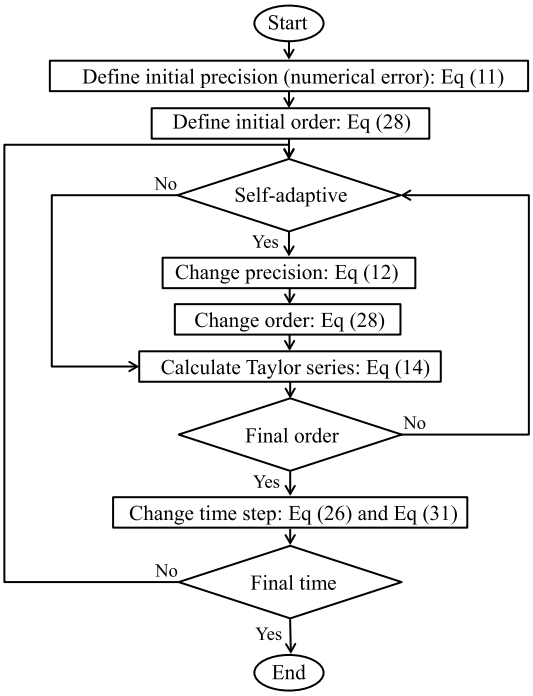

Fig. 1. Procedure of the self-adaptive CNS algorithm.

In a chaotic system, the averaged level of numerical noise should increase exponentially within a temporal interval $[0, T_c]$, i.e.

$$\varepsilon(t) = \varepsilon_0 \exp(\kappa t), \quad 0 \le t \le T_c, \tag{3}$$

where the noise growing exponent κ is the maximum Lyapunov exponent. Then, the critical predictable time T_c can be estimated by:

$$T_c = \frac{1}{\kappa} \ln(\frac{\varepsilon_c}{\varepsilon_0}), \tag{4}$$

where ε_c denotes the critical numerical noise (numerical noise at critical time T_c). It is important to note that each CNS result $x(t)$ is rigorously compared with another CNS result $x'(t)$, obtained using a higher-order Taylor expansion and a greater number of significant digits. This comparison ensures that the difference between the two results remains smaller than the required precision over the entire computation time, $t \in [0, T_c]$.

2.2. Dealing with the equations

In this subsection, we will briefly review the approach employed in [14] for solving the Lorenz-63 equations by CNS, the Lorenz-63 equations are defined by

$$\begin{aligned} \dot{x_1}(t) &= \sigma(x_2(t) - x_1(t)), \\ \dot{x_2}(t) &= R x_1(t) - x_2(t) - x_1(t) x_3(t), \\ \dot{x_3}(t) &= x_1(t) x_2(t) - b x_3(t), \end{aligned} \tag{5}$$

where σ, R, b are physical parameters, $x_1(t)$, $x_2(t)$, $x_3(t)$ are variables of the Lorenz 63 system and the dot denotes the differentiation with respect to the time t.

To obtain a N-th order Taylor series iterative formulae, we differentiate both sides of the Lorenz equations $(N-1)$ times with respect to t and then dividing them by $n!$, and have

$$\begin{aligned} x_1^{[n+1]} &= \frac{\sigma}{n+1}(x_2^{[n]} - x_1^{[n]}), \\ x_2^{[n+1]} &= \frac{1}{n+1}(R x_1^{[n]} - x_2^{[n]} - \sum_{j=0}^{n} x_1^{[j]} x_3^{[n-j]}), \\ x_3^{[n+1]} &= \frac{1}{n+1}(b x_1^{[n]} + \sum_{j=0}^{n} x_1^{[j]} x_2^{[n-j]}), \end{aligned} \tag{6}$$

where $x_1^{[n]}, x_2^{[n]}, x_3^{[n]}$ are the coefficients of the Taylor series

$$\begin{aligned} x_1^{[n]} &= \frac{1}{n!}\frac{\partial^n x_1(t)}{\partial t^n}, \\ x_2^{[n]} &= \frac{1}{n!}\frac{\partial^n x_2(t)}{\partial t^n}, \\ x_3^{[n]} &= \frac{1}{n!}\frac{\partial^n x_3(t)}{\partial t^n}. \end{aligned} \tag{7}$$

2.3. Self-adaptive order and time step size

The numerical noise is defined by the maximum of the truncation error and round-off error. So, Qin and Liao [38] let the truncation error is at the same level as the round-off error. And they use the optimal time step size, which is given by Barrio [41]

$$\Delta t = \min\left(\frac{\mathrm{tol}^{\frac{1}{N}}}{\left\| x_i^{[N-1]}(t) \right\|_\infty^{\frac{1}{N-1}}}, \frac{\mathrm{tol}^{\frac{1}{N+1}}}{\left\| x_i^{[N]}(t) \right\|_\infty^{\frac{1}{N}}} \right), \tag{8}$$

where tol denotes the allowed tolerance and $\|\cdot\|_\infty$ denotes the supremum norm. The order of Taylor expansion for parallel technology is given by an empirical formula

$$N = \lceil -\mu \log_{10}(\mathrm{tol}) \rceil = \lceil \mu N_s \rceil, \tag{9}$$

where the tol is using the round-off error form $\mathrm{tol} = 10^{-N_s}$ and N_s denotes the number of significant digits. A linear ratio, μ, is present in (9) and was empirically set to 1.5 by Qin and Liao [38].

2.4. Self-adaptive number of significant digits

As noted in (3), numerical noise grows exponentially over time. Consequently, for a larger critical time T_c, it becomes necessary to employ a sufficiently small initial numerical noise threshold ε_0. Assuming $\varepsilon_0 = 10^{-N_s}$, the equation (3) has the form

$$\varepsilon(t) = \varepsilon_0 \exp(\kappa t) = 10^{-\left(N_s - \kappa t / \ln 10\right)}, \quad t \in \left[0, T_c\right], \tag{10}$$

and the number of significant digits N_s at time T_c has the relationship

$$N_s = \left\lceil \frac{\gamma \kappa T_c}{\ln 10} - \log_{10} \varepsilon_c \right\rceil, \tag{11}$$

where $\gamma \ge 1$ denotes a safety factor in the adaptive number of significant digits implementation, it is set to be $\gamma = 1$ in the fix number of significant digits implementation [14,38]. Since holding the large N_s significantly reduces computational efficiency, and it is unnecessary to keep the N_s during the whole integration, the number of significant digits N_s can decrease with time

$$N_s = \left\lceil \frac{\gamma \kappa \left(T_c - t'\right)}{\ln 10} - \log_{10} \varepsilon_c \right\rceil, \tag{12}$$

where $t' = n\Delta T$ with $n = 0, 1, 2, \ldots$ and ΔT is the changing interval of the number of significant digits. As noted in [38], the approach described in (12) should not be applied to the entire computation. Instead, it is necessary to implement a stabilization period, T_r, prior to the time approaching T_c i.e. the value of N_s is stopped decreasing after $T_c - T_r$.

3. Implementation in CNSPy

In this section, we will show the implementation in CNSPy, including automatic differentiation (AD), arbitrary number of significant digits and a new parallel optimal order formula.

3.1. Automatic differentiation

Differentiating equations (e.g., (5) and (6)) to derive iterative formulae is the most straightforward approach. However, this process is challenging to automate. Automatic differentiation (AD) provides an efficient method to automatically generate the iterative formulae for Taylor series, and it has been implemented in various software packages, such as taylor[37], TIDES[40], and TaylorIntegration.jl[39], which employ the Taylor series method to solve ordinary differential equations (ODEs). Although the foundational concept of AD has been thoroughly explained in [37], we include it here for completeness.

Assuming the objective ODEs

$$\dot{\mathbf{x}} = f(\mathbf{x}, t), \tag{13}$$

and the function $f(\mathbf{x}, t)$ is consisted of basic binary arithmetic $\mathfrak{b}(\cdot)$, such as $+$, $-$, $\times$, $\div$, exp, log, sin, cos, etc., then the function (13) can be decomposed to S binary operation $\mathbf{f} = \{f_s, s = 0, 1, 2, \ldots, S\}$, i.e.

$$\begin{aligned} f_1 &= \mathfrak{b}_1(x_i, x_j), && x_i, x_j \in \mathbf{x} \\ &\cdots \\ f_{m+1} &= \mathfrak{b}_2(x_i, f_s), && x_i \in \mathbf{x} \text{ and } s < m \\ &\cdots \\ f_S &= \mathfrak{b}_S(f_{s_1}, f_{s_2}), && s_1, s_2 < S \\ f(\mathbf{x}, t) &= f_S, \end{aligned} \tag{14}$$

where m denotes the number of binary operation between $\mathbf{x}$. The structure depicted in equation (14) represents the parse tree corresponding to equation (13). To solve equation (13) using the Taylor series method, the binary operation $\mathfrak{b}(\cdot)$ must be transformed into the fundamental Taylor series binary arithmetic $\mathfrak{B}(\cdot)$. By providing all the essential iterative arithmetic formulae for $\mathfrak{B}(\cdot)$, equation (13) can be efficiently in an order-by-order manner, following a procedure analogous to equation (14).

The $\mathfrak{B}(\cdot)$ can be easily derived by Leibniz rule

$$a^{[n]}(t) = \sum_{j=0}^{n} b^{[j]}(t) c^{[n-j]}(t), \tag{15}$$

where $a^{[n]}, b^{[n]}, c^{[n]}$ are the coefficients of the Taylor series similar to equation (7). It should be noted that all of $\mathfrak{B}(\cdot)$ consist a summation $\sum^{n}$. This will lead to a $\mathcal{O}(N^2)$ computational effort (or number of arithmetic operations), since $\mathfrak{B}(\cdot)$ must do N times in every time step.

For simplicity of use and ultimate computational efficiency, we will define the ODEs in Python using an elegant symbolic mathematics package Sympy[42], and translate the symbolic description into a set of C files to run efficiently.

3.2. Arbitrary number of significant digits implementation

There are many methods to perform the arbitrary number of significant digits arithmetical operation, the most famous one is GNU MPFR Library [43]. It is designed in the C language and has been implemented in other languages, such as mpreal in C++, bigfloat package in Python and BigFloat.jl in Julia. It is well known that the language C has the best performance of all other languages, and the arbitrary number of significant digits in other language is based on the C version of GNU MPFR. So, it is natural to use the GNU MPFR in C language in CNSPy for speed consideration. However, the worrisome of using C language is the procedure-oriented feature and can not overload operators as in object-oriented language such as Cpp. To make the code readable and maintain the computational efficient, we use a series of macro define of the binary operation $\mathfrak{B}(\cdot)$, which will be substituted during the compile.

The self-adaptive number of significant digits implementation in CNSPy using equation (12). The change of the number of significant digits can be realized by the function mpfr_prec_round in GNU MPFR. The mpfr_prec_round function will cost the extra time during the calculation, so it is unnecessary to change the number of significant digits every time step [38].

3.3. Optimal order for parallel implementation

As we review previously, the optimal order of self-adaptive CNS is given by equation (9), which is originally from [37]

$$N = \left\lceil -\frac{1}{2}\ln\left(\frac{\mathrm{tol}}{M_m}\right) + 1 \right\rceil \approx \lceil 1.15 N_s \rceil, \tag{16}$$

where M_m is a positive constant and can be chosen to be 1. Equation (16) is derived under the assumption that the computational effort of the Taylor Series Method scales as $\mathcal{O}(N^2) = c(N+1)^2$, representing the serial computational cost. Equation (9) builds upon this relationship, incorporating an empirical coefficient of approximately 1.5 for the ratio, determined through comparisons of various μ values in equation (9). A theoretical derivation of the optimal value for the parameter μ in the context of parallel implementation is necessary to enhance understanding and efficiency.

For a parallel implementation, the primary component amenable to parallelization is the summation term in the equation (15):

$$a_{\mathrm{id}}^{[n]}(t) = \sum_{\substack{j=\mathrm{id} \\ \text{step } n_p}}^{n} b^{[j]}(t) c^{[n-j]}(t), \tag{17}$$

where $\mathrm{id} = 0, 1, 2, \ldots, n_p$ represents the identifier of the processor, and n_p denotes the total number of the processors.

Then, the summation of $a_{\mathrm{id}}^{[n]}(t)$ is

$$a^{[n]}(t) = \sum_{j=0}^{n_p} a_j^{[n]}(t). \tag{18}$$

The computational effort of (17) and (18) is $\mathcal{O}_p(N/n_p + \log_2(n_p))$. Hence, the total computational effort is

$$\begin{aligned} \sum_{j=0}^{N} \mathcal{O}_p\left(\frac{j}{n_p} + \log_2(n_p)\right) &= \mathcal{O}_p\left(\frac{N^2}{n_p} + N\log_2(n_p)\right) \\ &= c\left(\frac{(N+1)^2}{n_p} + (N+1)\log_2(n_p)\right). \end{aligned} \tag{19}$$

As discussed in Reference [44], for a given time step $h = t - t_m$ and time t_m, if Δt is small enough, the solution of (13) at point x_m has the form

$$x_m(t) = \sum_{j=0}^{\infty} x_m^{[j]}(t_m)(h)^j. \tag{20}$$

With a sufficiently small $\Delta t = t_{m+1} - t_m$ and a sufficiently large N, the (20) is truncated to (13) at point x_m has the form

$$x_{m+1} = \sum_{j=0}^{N} x_m^{[j]}(t_m)(\Delta t)^j, \tag{21}$$

satisfy

$$\left\| x_m\left(t_{m+1}\right) - x_{m+1} \right\| \le \varepsilon. \tag{22}$$

Assume that series (20) is analytic on a disk of radius ρ_m, and there exists a positive constant M_m such that

$$\left|x_m^{[j]}\right| \le \frac{M_m}{\rho_m^j}, \quad \forall j \in \mathbb{N}. \tag{23}$$

The truncation error of (21) can be written as

$$\mathrm{tol} = x_{m+1}^{[N+1]}(t_m)(\Delta t)^{N+1}. \tag{24}$$

So, the tolerance tol introduced from (21) can be given by

$$\mathrm{tol} \approx M_m \left(\frac{\Delta t}{\rho_m}\right)^{N+1}. \tag{25}$$

Then, to satisfy the required tolerance tol, the time step Δt should be

$$\Delta t \approx \rho_m \left(\frac{\mathrm{tol}}{M_m}\right)^{\frac{1}{N+1}}. \tag{26}$$

If the required accuracy ε tends to 0, the values of Δt and N that give the required accuracy can be found by minimizing the global number of operations. Thus, together with (19), the computational effort per time step can be given by

$$\phi(N) = \frac{c\left(\frac{(N+1)^2}{n_p} + (N+1)\log_2(n_p)\right)}{\rho_m \left(\frac{\mathrm{tol}}{M_m}\right)^{\frac{1}{N+1}}}. \tag{27}$$

We can solve $\phi'(N) = 0$ to find a N such that $\phi(N)$ is the minimum value, and we have

$$\begin{aligned} N = &-\sqrt{\frac{1}{16}\left(\ln(\frac{\mathrm{tol}}{M_m}) - n_p\log_2(n_p)\right)^2 - \frac{1}{4}n_p\log_2(n_p)\ln(\frac{\mathrm{tol}}{M_m}) +} \\ &- \frac{1}{4}\ln(\frac{\mathrm{tol}}{M_m}) - n_p\log_2(n_p) - 1. \end{aligned} \tag{28}$$

It should be noted that when $n_p = 1$, (28) become

$$N = -\frac{1}{2}\ln(\frac{\mathrm{tol}}{M_m}) - 1, \tag{29}$$

which is consistent with Simó's conclusion [44].

Following [37], we choose $M_m = 1$, and radius of convergence ρ_m is given by

$$\rho_m = \min\{\rho_m^{(N-1)}, \rho_m^{(N)}\}, \tag{30}$$

where $\rho_m^{(j)}$ is given by

$$\begin{aligned} \rho_m^{(j)} &= \left(\frac{1}{\left\|x_m^{[j]}\right\|_\infty}\right)^{\frac{1}{j}}, \quad 1 \le j \le p \text{ and } \|x_m\|_\infty \le 1, \\ \rho_m^{(j)} &= \left(\frac{\|x_m\|}{\left\|x_m^{[j]}\right\|_\infty}\right)^{\frac{1}{j}}, \quad 1 \le j \le p \text{ and } \|x_m\|_\infty > 1. \end{aligned} \tag{31}$$

Now, it is natural to ask for the value of μ from (9). The (28) can be written as

$$N = N(\mathrm{tol}) - n_p\log_2(n_p) - 1. \tag{32}$$

Since the value of ln(tol) is negative, it is obvious that the tol related term $N(\mathrm{tol})$ has

$$\begin{aligned} N(\mathrm{tol}) > -\sqrt{\frac{1}{16}\left(\ln(\frac{\mathrm{tol}}{M_m})\right)^2} - \frac{1}{4}\ln(\frac{\mathrm{tol}}{M_m}) = -\frac{1}{4}\ln(\frac{\mathrm{tol}}{M_m}) - \frac{1}{4}\ln(\frac{\mathrm{tol}}{M_m}) \\ \approx 1.15N_s. \end{aligned} \tag{33}$$

If $N(\mathrm{tol})$ follows a linear form proportional to μN_s, then the ratio μ must satisfy $\mu > 1.15$. This indicates that a larger value of μ is required when employing a parallel implementation, aligning with the value of 1.5 selected by Qin and Liao [38]. However, the value of μ may vary depending on the total number of processors n_p, and the relationship defining $N(\mathrm{tol})$ may deviate from linearity.

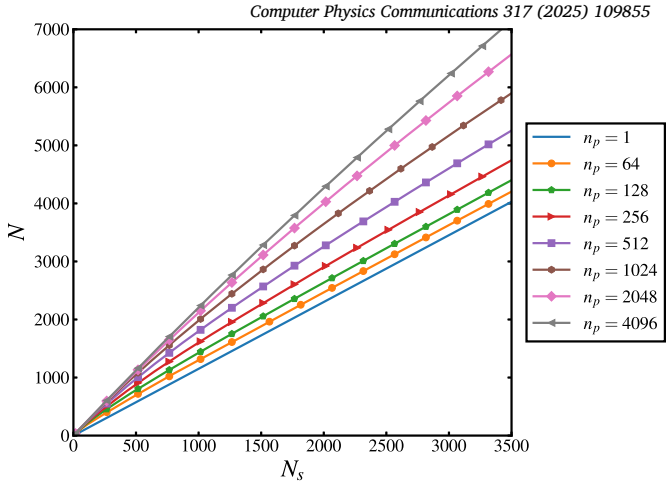

Fig. 2. Relationship between number of significant digits N_s and optimal order N with different number of processor n_p. $n_p = 1$ is the serial implementation, which has the linear relation (16).

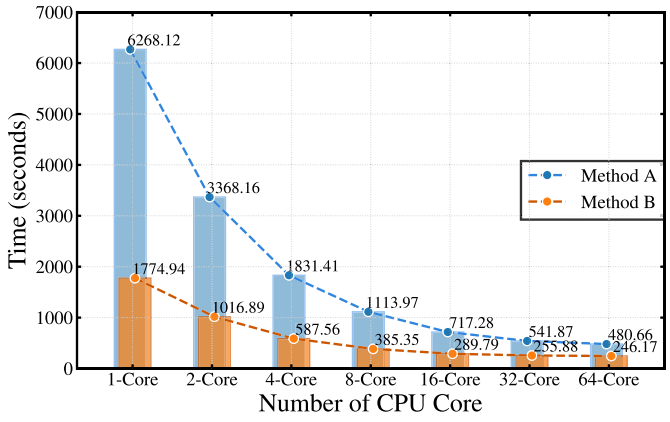

Fig. 3. Computation time comparison under different parallel configurations. Method A represents the conventional Taylor series algorithm with adaptive step size (26) and adaptive order selection (9). Method B demonstrates the CNSPy adaptive algorithm combining step size control (26), adaptive order selection (28), and significant digit adjustment (11). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

As illustrated in Fig. 2, the linear relationship between N and N_s progressively decreases with increasing number of processors n_p. When applying linear regression to estimate the value of μ between N and N_s, μ increases as n_p increases. This trend is reasonable since the computational effort for a single time step is directly influenced by the ratio N/n_p. With more processors, a higher order N can be used to reduce error, allowing for a larger time step Δt, thereby reducing the total number of calculations. Furthermore, it becomes evident that a linear relationship cannot accurately capture the connection between N and N_s when n_p is sufficiently large. To address this, `CNSPy` numerically solves (28) and employs a quadratic regression to derive a surrogate formula. This approach involves less computational effort than directly solving (28), while preserving sufficient accuracy in approximating the optimal relationship.

3.4. Computational efficiency analysis

To quantitatively evaluate the computational efficiency improvement of the `CNSPy` adaptive algorithm, this section compares the perfor-

mance between conventional Taylor series software algorithms and the CNSPy adaptive algorithm under different parallel core configurations. For representative analysis, we employ the classical Lorenz-63 chaotic dynamical system as our numerical example. The computational time required for simulating 1000 Lorenz unit time is measured under serial (1 core) and parallel configurations (2, 4, 6, 8, 16, 32, and 64 cores).

The conventional Taylor series algorithm implements adaptive step size control (26) and adaptive order selection (9), while the CNSPy adaptive algorithm integrates three adaptation mechanisms: step size control (26), adaptive order selection (28), and significant digit adjustment (11).

Fig. 3 illustrates the computational time scaling characteristics, where blue bars and trend lines depict the conventional algorithm's performance, while orange components represent CNSPy's results. Both methods show decreasing computation times with increasing core counts, but CNSPy consistently outperforms the conventional approach across all configurations.

The CNSPy algorithm demonstrates significant time reduction, particularly in serial computation (1-core). While the performance gap narrows with higher core counts due to Amdahl's law limitations on parallel speedup, the conventional method still requires twice the computation time of CNSPy at 64 cores. This substantial improvement confirms that the CNSPy adaptive algorithm provides an efficient computational framework for clean numerical simulation of chaotic dynamical systems through its multi-adaptation mechanism.

4. Numerical experiments

In this section, we will test CNSPy via some typical models: Lorenz-63 model (the simplest chaotic model), Lorenz-96 model (a N-dimension chaotic model) and three-body problem. The numerical experiments are run on Intel(R) Xeon(R) Gold 6148 with 64 CPUs on our local cluster.

The definition of the relative error ε_r is

$$\varepsilon_r(t) = \frac{\sum_{i=1}^{N_d} \left| x_i'(t) - x_i(t) \right|}{\sum_{i=1}^{N_d} \left| x_i'(t) \right|}, \tag{34}$$

where N_d is the number of the variable (i.e. dimensions of the model), $x_i'(t)$ and $x_i(t)$ are two CNS results with different level of the background numerical noise, respectively.

Usually, the $x_i'(t)$ represents a CNS result with lower level of the background numerical noise than the $x_i(t)$. To compare the results, we will calculate the (5) with different settings. The adaptive setting is using the self-adaptive CNS algorithm via CNSPy, which is the default setting and recommended setting in CNSPy. The fix setting is using the fix order and precision via CNSPy. The examine setting is using the same setting as the fix setting but with a higher order and precision, i.e. $x_i'(t)$. The RKwD setting is using the fourth-order Runge-Kutta method with double precision via `quick_run()` function in CNSPy.

4.1. Lorenz-63 model

First of all, let us consider the famous Lorenz-63 model as illustrated in (5). The total calculated time is $T_t = 1000$ (Lorenz unit time) and the Lyapunov exponent is $\kappa = 0.91$. The parameters in (5) are

$$\sigma = 10,\ R = 28,\ b = \frac{8}{3}, \tag{35}$$

with the initial condition

$$x_1(0) = -15.8,\ x_2(0) = -17.48,\ x_3(0) = 35.64. \tag{36}$$

Under the above settings, the Lorenz-63 model exhibits chaotic property and has a strange attractor.

The adaptive parameters in CNSPy (adaptive) are given by

$$\gamma = 1.1,\ T_r = 300,\ \Delta T = 50, \tag{37}$$

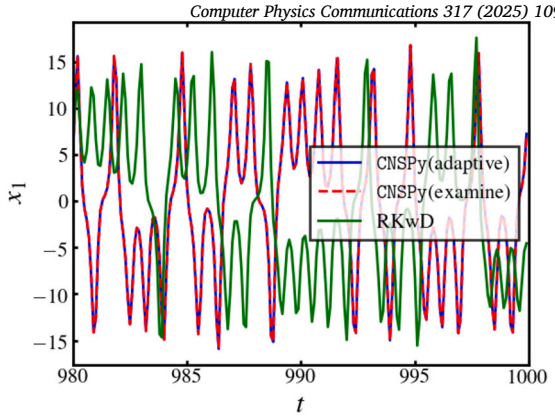

Fig. 4. Time histories of x_1 in Lorenz-63 model calculated by CNSPy (adaptive in time step, order and number of significant digits), CNSPy (fix in time step, order and number of significant digits) and RKwD (fourth-order Runge-Kutta method with double precision).

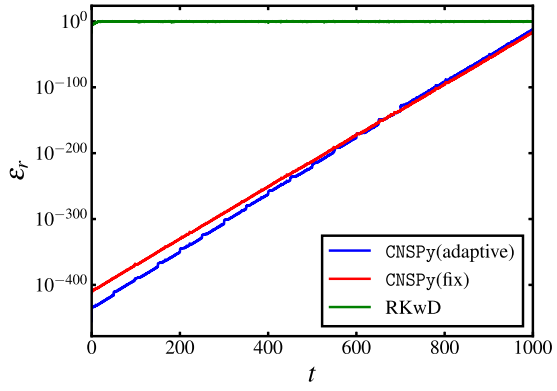

Fig. 5. Relative error ε_r growth of the Lorenz-63 model solutions calculated by CNSPy (adaptive in time step, order and number of significant digits), CNSPy (fix in time step, order and number of significant digits) and RKwD (fourth-order Runge-Kutta method with double precision).

to give a good performance of CNS calculation according to reference [38].

Here, we use the CNSPy with fix setting to recover the CNS calculation in reference [14,18]. The critical time T_c are set to be the same as total calculated time T_t. To use CNS gain the reliable numerical solution, one must give the order M of the Taylor series and the N_s of the number of the significant digits in the fix setting. By using (11) and (9), the order is $M = 617$ and the precision is $N_s = 411$ with the critical numerical noise $\varepsilon_c = 10^{-16}$. To check the solution's numerical error ε, the parameters of examine setting are $M = 800$ and $N_s = 500$. As shown in Fig. 4, the time histories of x_1 calculated by CNSPy (adaptive), CNSPy (examine) and RKwD are given in the last 20 lorenz unit time. The x_1 of recommended setting CNSPy (adaptive) is exactly the same as the examine setting CNSPy (examine), which means the results given by CNSPy are reproducible, while the x_1 of RKwD is completely different from the reproducible results given by CNSPy.

Now, let us focus on the error of the whole calculation process. As shown in Fig. 5, the relative error ε_r of CNSPy (adaptive), CNSPy (fix) and RKwD are given in the temporal interval $t \in [0, T_t]$. The relative error ε_r of the RKwD setting is basically at the level of 1, which means the magnitude of the numerical noise is same as the 'true' physical solution. If we use the (11) with the double precision ($N_s = 15$ and $\varepsilon_c = 10^{-2}$), the critical time of the RKwD can be easily approximated, i.e. $T_c \approx 33$ (Lorenz unit time). Thus, the relative error ε_r grows exponential to about 10^0 within $t \in [0, 33]$ (Lorenz unit time). The relative error ε_r of the CNSPy (adaptive) and CNSPy (fix) are all maintained far below the level of 10^0. It should be noted that the relative error ε_r is not the same at the

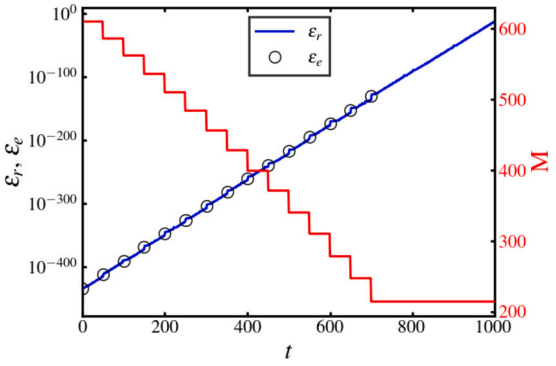

Fig. 6. Relative error ε_r growth of the Lorenz-63 model solution calculated by adaptive CNSPy (blue solid line referenced to left axis), solutions' expected error ε_e given by CNSPy's adaptive algorithm (black symbol referenced to left axis) and the time historical order given by CNSPy (red solid line referenced to right axis).

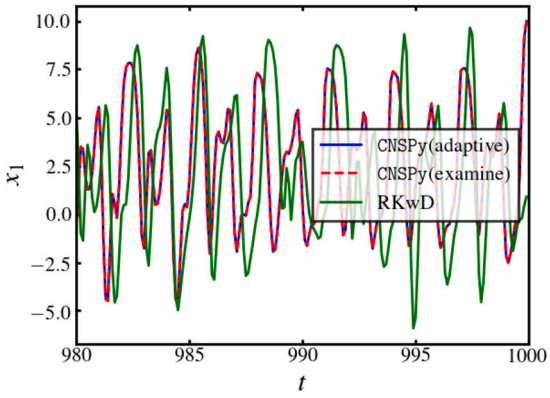

Fig. 7. Time histories of x_1 in Lorenz-96 model ($N_d = 5$) calculated by CNSPy (adaptive in time step, order and number of significant digits), CNSPy (fix in time step, order and number of significant digits) and RKwD (fourth-order Runge-Kutta method with double precision).

beginning of CNSPy (adaptive) and CNSPy (fix). This difference comes from the safety factor in (12), to balance the extra numerical noise that introduces from the adaptive algorithm.

To further check the adaptive algorithm in CNSPy, the relative error ε_r, expected error ε_e that CNSPy gives to the calculation and the order M during the calculation are given in Fig. 6. The relative error ε_r changes as the CNSPy designed, at each ΔT, the order M decreases and the expected error ε_e increases to lead the relative error ε_r. At the last remained interval $t \in [T_c - T_r, T_c]$, the order M stops to decrease and the relative error ε_r grows naturally to the final time. This is the simplest example to use CNSPy solve a chaotic system.

4.2. Lorenz-96 model

The Lorenz-96 model is defined as follows:

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})\, x_{i-1} - x_i + F, \quad i = 1, \ldots, N_d, \tag{38}$$

where $x_{-1} = x_{N-1}$, $x_0 = x_{N_d}$, $x_1 = x_{N_d+1}$ and $N_d \geq 4$. $F = 8$ is a common value that causes chaos. It is a N-dimensional chaotic system and is convenient to solve by CNSPy. In this subsection, we use $N_d = 5$, and the Lyapunov exponent is $\kappa = 0.48$. The initial condition is

$$x_1(0) = 8.01,\ x_2(0) = 8.0,\ x_3(0) = 8.0,\ x_4(0) = 8.0,\ x_5(0) = 8.0, \tag{39}$$

and the adaptive parameters in CNSPy (adaptive) are given by

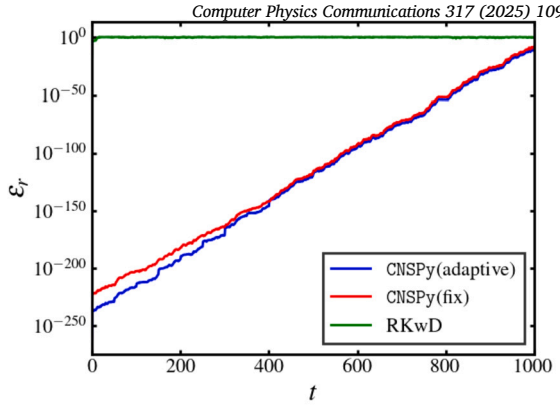

Fig. 8. Relative error ε_r growth of the Lorenz-96 model ($N_d = 5$) solutions calculated by CNSPy (adaptive in time step, order and number of significant digits), CNSPy (fix in time step, order and number of significant digits) and RKwD (fourth-order Runge-Kutta method with double precision).

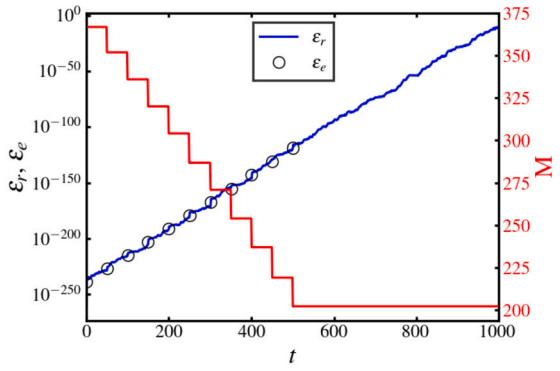

Fig. 9. Relative error ε_r growth of the Lorenz-96 model ($N_d = 5$) solution calculated by adaptive CNSPy (blue solid line referenced to left axis), solutions' expected error ε_e given by CNSPy's adaptive algorithm (black symbol referenced to left axis) and the time historical order given by CNSPy (red solid line referenced to right axis).

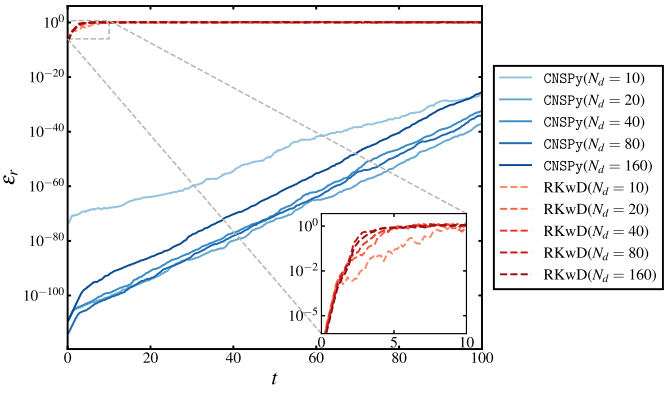

Fig. 10. Relative error ε_r growth of the Lorenz-96 model with different N_d calculated by adaptive CNSPy (blue solid line) and RKwD (red dash line).

$$\gamma = 1.15,\ T_r = 500,\ \Delta T = 50, \tag{40}$$

to fit the Lorenz-96 model perfectly.

Similarly to the previous subsection, we contract the CNSPy default settings, i.e. adaptive settings with the fix settings and examine settings. The order and precision are set to $M = 338$ and $N_s = 225$ in fix settings and $M = 400$ and $N_s = 300$ in examine settings according to (9) and (11), respectively. As shown in Fig. 7, the time histories of x_1 calculated

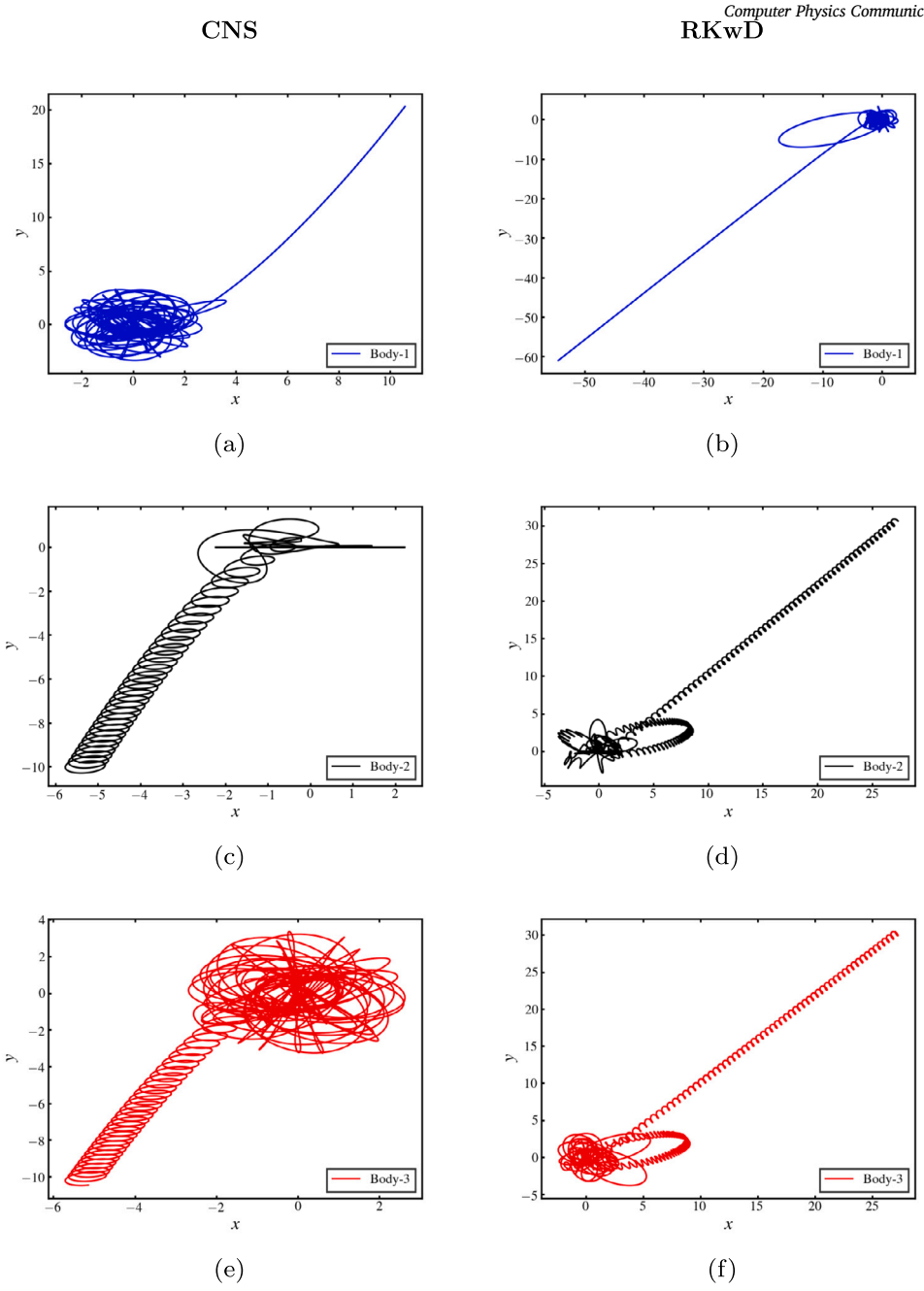

Fig. 11. Comparisons of the trajectories of the three-body system in $x-z$ given by the CNS (left) and the RKwD (right) during $t \in [0, 1000]$. (a)-(b): $x-z$ trajectories of Body-1; (c)-(d): $x-z$ trajectories of Body-2; (e)-(f): $x-z$ trajectories of Body-3.

by CNSPy (adaptive), CNSPy (examine) and RKwD are given in the last 20 lorenz unit time. The x_1 of recommended setting CNSPy (adaptive) is exactly the same as the examine setting CNSPy (examine), which means the results given by CNSPy are reproducible, while the x_1 of RKwD is completely different from the reproducible results given by CNSPy. Then, the relative error of the whole calculation process is considered, as shown in Fig. 8 and Fig. 9. The CNSPy provide a good control of the numerical noise, and both CNSPy (adaptive) and CNSPy (fix) maintain the relative error ε_r far below the level of 10^0.

To further demonstrate the utility of CNSPy in high-dimensional chaotic systems, Fig. 10 presents a comparison of the relative errors ε_r produced by the proposed CNSPy (adaptive) and the traditional RKwD when solving the Lorenz96 model for varying system dimensions N_d. It is evident that CNSPy consistently maintains low relative errors across all tested dimensions, including challenging high-dimensional cases such as $N_d = 160$. In contrast, the relative error ε_r associated with RKwD increases rapidly with both system dimension and integration time, highlighting the limitations of conventional methods in achieving reliable long-term integration for high-dimensional chaotic systems. These results underscore the reliability of CNSPy for long-term simulations in complex dynamical regimes.

4.3. Three-body model

The famous three-body model is defined as follows:

$$\ddot{x}_{k,i} = \sum_{j=1, j\neq i}^{3} \rho_j \frac{x_{k,j} - x_{k,i}}{R_{i,j}^3}, \quad k = 1, 2, 3, \tag{41}$$

where $x_{k,i}$ is the i-th body along k-th direction,

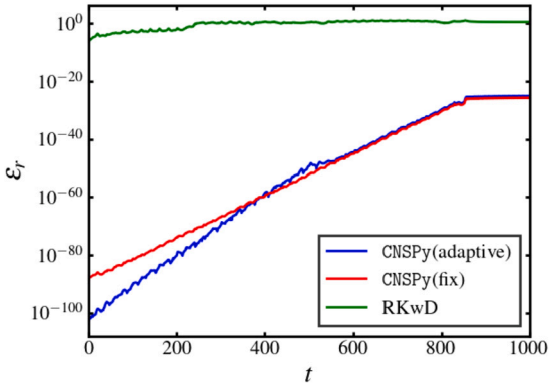

Fig. 12. Relative error ε_r growth of the three-body model solutions calculated by CNSPy (adaptive in time step, order and number of significant digits), CNSPy (fix in time step, order and number of significant digits) and RKwD (fourth-order Runge-Kutta method with double precision).

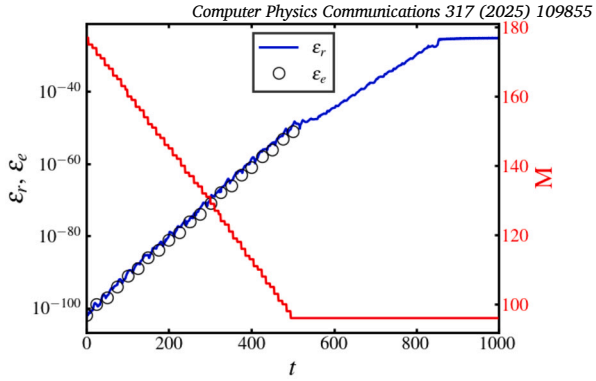

Fig. 13. Relative error ε_r growth of the three-body model solution calculated by adaptive CNSPy (blue solid line referenced to left axis), solutions' expected error ε_e given by CNSPy's adaptive algorithm (black symbol referenced to left axis) and the time historical order given by CNSPy (red solid line referenced to right axis).

$$R_{i,j} = \left[\sum_{k=1}^{3} \left(x_{k,j} - x_{k,i}\right)^2\right]^{\frac{1}{2}} \tag{42}$$

denotes the distance between the i-th body and the j-th body, and

$$\rho_i = \frac{m_i}{m_1}, \quad i = 1,2,3 \tag{43}$$

denotes the ratio of mass, where m_i denotes the mass of the i-th body. Following Liao [17], the initial positions are

$$\mathbf{r}_1 = (0,0,-1) + d\mathbf{r}_1, \quad \mathbf{r}_2 = (0,0,0), \quad \mathbf{r}_3 = -\left(\mathbf{r}_1 + \mathbf{r}_2\right), \tag{44}$$

and the initial velocities are

$$\dot{\mathbf{r}}_1 = (0,-1,0), \quad \dot{\mathbf{r}}_2 = (1,1,0), \quad \dot{\mathbf{r}}_3 = -\left(\dot{\mathbf{r}}_1 + \dot{\mathbf{r}}_2\right), \tag{45}$$

where $d\mathbf{r}_1 = \delta(1,0,0))$ is the micro-level physical uncertainty with $\delta = 10^{-60}$. The mass of the bodies are all equal to each other, i.e. $\rho_i = 1$ with $i = 1,2,3$ and the Lyapunov exponent is $\kappa = 0.168$. The adaptive parameters in CNSPy (adaptive) are given by

$$\gamma = 1.4,\ T_r = 500,\ \Delta T = 5, \tag{46}$$

to fit the three-body model perfectly. The order and precision are set to $M = 135$ and $N_s = 90$ in fix settings and $M = 220$ and $N_s = 150$ in examine settings according to (9) and (11), respectively.

In Fig. 11, we present the trajectories of the three-body system computed using CNSPy (adaptive) and RKwD over the entire simulation period. The trajectories produced by CNSPy align precisely with the results reported by Liao [17], affirming the reproducibility of the Clean Numerical Simulation (CNS) method and the accuracy of CNSPy. Conversely, the trajectories calculated using RKwD exhibit significant deviations, with the three bodies diverging markedly due to numerical noise. Regarding the relative error ϵ_r, as illustrated in Figs. 12 and 13, both CNSPy (adaptive) and CNSPy (fixed) effectively control the relative error, demonstrating robust performance in maintaining accuracy. In contrast, RKwD fails to achieve comparable precision.

5. Conclusion

In this paper, we have presented and analyzed the implementation of the CNS method in the context of chaotic dynamical systems, specifically focusing on the solution of ordinary differential equations (ODEs) through Taylor series method. The primary objective was to address the challenge of numerical noise, which grows exponentially in chaotic systems, by introducing a self-adaptive approach within the CNSPy.

Our results demonstrate that CNSPy effectively reduces numerical noise by leveraging higher-order Taylor expansions and extended precision formats. This reduction in error enables for more accurate long-term predictions, providing a reliable solution to chaotic systems where traditional numerical methods might fail due to accumulated numerical noise. Through careful analysis, we established that the ratio μ in the parallel implementation must be adjusted based on the number of processors n_p. This implementation gives better performance than conventional Taylor series algorithm under serial (1 core) and parallel configurations (2, 4, 6, 8, 16, 32, and 64 cores).

The application of CNSPy was validated by calculating the solutions of several chaotic systems and the performance of controlling the numerical noise is shown by comparing the CNSPy with RKwD. The results showed that CNSPy produced highly accurate and reproducible results, consistent with previous studies, while RKwD exhibited significant deviations due to numerical noise. CNSPy offers a robust, self-adaptive framework for solving chaotic ODEs, providing a reliable tool for both theoretical and computational studies of dynamical systems.

Finally, from the viewpoint of user, it should be emphasized that, the software automates the CNS computational workflow by automatically converting Python-defined ODEs into a parallelized C code, eliminating the need for error-prone manual derivation and coding while ensuring high efficiency in high-performance computing (HPC) environments.

CRediT authorship contribution statement

Bo Zhang: Writing – original draft, Visualization, Validation, Software. **Shijun Liao:** Writing – review & editing, Supervision, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We express our sincere thanks to the anonymous reviewers for their valuable comments and suggestions. This work is partly supported by National Natural Science Foundation of China (No. 12272230).

Data availability

The source code of the package CNSPy described in the paper is free and available at https://github.com/sjtu-liao/cnspy.

References

[1] H. Poincaré, Sur le problème des trois corps et les équations de la dynamique, Acta Math. 13 (1) (1890) A3–A270.

[2] E.N. Lorenz, Deterministic nonperiodic flow, J. Atmos. Sci. 20 (2) (1963) 130–141.
[3] E.N. Lorenz, Computational chaos-a prelude to computational instability, Phys. D, Nonlinear Phenom. 35 (3) (1989) 299–317.
[4] B.-W. Shen, R.A. Pielke Sr, X. Zeng, J.-J. Baik, S. Faghih-Naini, J. Cui, R. Atlas, Is weather chaotic?: coexistence of chaos and order within a generalized Lorenz model, Bull. Am. Meteorol. Soc. 102 (1) (2021) E148–E158.
[5] B.-W. Shen, A review of Lorenz's models from 1960 to 2008, Int. J. Bifurc. Chaos 33 (10) (2023) 2330024.
[6] E.N. Lorenz, Computational periodicity as observed in a simple system, Tellus, Ser. A Dyn. Meteorol. Oceanogr. 58 (5) (2006) 549–557.
[7] N. Chandramoorthy, Q. Wang, On the probability of finding nonphysical solutions through shadowing, J. Comput. Phys. 440 (2021) 110389.
[8] T.E. Nazaré, E.G. Nepomuceno, S.A. Martins, D.N. Butusov, A note on the reproducibility of chaos simulation, Entropy 22 (9) (2020) 953.
[9] J. Teixeira, C.A. Reynolds, K. Judd, Time step sensitivity of nonlinear atmospheric models: numerical convergence, truncation error growth, and ensemble design, J. Atmos. Sci. 64 (1) (2007) 175–189.
[10] K.M. Owolabi, B. Karaagac, Chaotic and spatiotemporal oscillations in fractional reaction-diffusion system, Chaos Solitons Fractals 141 (2020) 110302.
[11] K.M. Owolabi, A. Atangana, Numerical simulations of chaotic and complex spatiotemporal patterns in fractional reaction–diffusion systems, Comput. Appl. Math. 37 (2018) 2166–2189.
[12] K.M. Owolabi, Mathematical analysis and numerical simulation of chaotic noninteger order differential systems with Riemann-Liouville derivative, Numer. Methods Partial Differ. Equ. 34 (1) (2018) 274–295.
[13] K.M. Owolabi, A. Atangana, Mathematical analysis and computational experiments for an epidemic system with nonlocal and nonsingular derivative, Chaos Solitons Fractals 126 (2019) 41–49.
[14] S. Liao, On the reliability of computed chaotic solutions of non-linear differential equations, Tellus, Ser. A Dyn. Meteorol. Oceanogr. 61A (4) (2009) 550–564.
[15] S. Liao, Clean Numerical Simulation, Chapman and Hall/CRC, 2023.
[16] S. Liao, On the numerical simulation of propagation of micro-level inherent uncertainty for chaotic dynamic systems, Chaos Solitons Fractals 47 (2013) 1–12.
[17] S. Liao, Physical limit of prediction for chaotic motion of three-body problem, Commun. Nonlinear Sci. Numer. Simul. 19 (3) (2014) 601–616.
[18] S. Liao, P. Wang, On the mathematically reliable long-term simulation of chaotic solutions of Lorenz equation in the interval [0, 10000], Sci. China, Phys. Mech. Astron. 57 (2014) 330–335.
[19] T. Hu, S. Liao, On the risks of using double precision in numerical simulations of spatio-temporal chaos, J. Comput. Phys. 418 (2020) 109629.
[20] S. Qin, S. Liao, A kind of Lagrangian chaotic property of the Arnold-Beltrami-Childress flow, J. Fluid Mech. 960 (2023) A15, https://doi.org/10.1017/jfm.2023.190.
[21] X. Li, S. Liao, More than six hundred new families of Newtonian periodic planar collisionless three-body orbits, Sci. China, Phys. Mech. Astron. 60 (2017) 1–7.
[22] X. Li, Y. Jing, S. Liao, Over a thousand new periodic orbits of a planar three-body system with unequal masses, Publ. Astron. Soc. Jpn. 70 (4) (2018) 64.
[23] X. Li, S. Liao, Collisionless periodic orbits in the free-fall three-body problem, New Astron. 70 (2019) 22–26.
[24] S. Liao, X. Li, Y. Yang, Three-body problem – from Newton to supercomputer plus machine learning, New Astron. 96 (2022) 101850.
[25] M. Šuvakov, V. Dmitrašinović, Three classes of Newtonian three-body planar periodic orbits, Phys. Rev. Lett. 110 (11) (2013) 114301.
[26] A. Hudomal, New periodic solutions to the three-body problem and gravitational waves, Master of Science thesis, Faculty of Physics, Belgrade University, 2015.
[27] X. Li, S. Liao, Discovery of 10,059 new three-dimensional periodic orbits of general three-body problem, arXiv:2508.08568, https://arxiv.org/abs/2508.08568, 2025.
[28] S. Liao, S. Qin, Ultra-chaos: an insurmountable objective obstacle of reproducibility and replication, Adv. Appl. Math. Mech. 14 (4) (2022) 799–815.
[29] B. Zhang, S. Liao, Ultra-chaos in a meandering jet flow, Phys. D, Nonlinear Phenom. 455 (2023) 133886.
[30] S. Qin, S. Liao, Large-scale influence of numerical noises as artificial stochastic disturbances on a sustained turbulence, J. Fluid Mech. 948 (2022) A7.
[31] S. Qin, Y. Yang, Y. Huang, X. Mei, L. Wang, S. Liao, Is a direct numerical simulation (DNS) of Navier-Stokes equations with small enough grid spacing and time-step definitely reliable/correct?, J. Ocean Eng. Sci. 9 (3) (2024) 293–310.
[32] S. Liao, S. Qin, Noise-expansion cascade: an origin of randomness of turbulence, J. Fluid Mech. 1009 (2025) A2, https://doi.org/10.1017/jfm.2025.140.
[33] S. Liao, S. Qin, Physical significance of artificial numerical noise in direct numerical simulation of turbulence, J. Fluid Mech. 1008 (2025) R2, https://doi.org/10.1017/jfm.2025.200.
[34] S. Qin, S. Liao, Clean numerical simulation (CNS) of three-dimensional turbulent Kolmogorov flow, arXiv:2507.11805, https://arxiv.org/abs/2507.11805, 2025.
[35] J.R. Scott, M.C. Martini, High-speed solution of spacecraft trajectory problems using Taylor series integration, J. Spacecr. Rockets 47 (1) (2010) 199–202.
[36] R. Barrio, Performance of the Taylor series method for ODEs/DAEs, Appl. Math. Comput. 163 (2) (2005) 525–545.
[37] À. Jorba, M. Zou, A software package for the numerical integration of ODEs by means of high-order Taylor methods, Exp. Math. 14 (1) (2005) 99–117.
[38] S. Qin, S. Liao, A self-adaptive algorithm of the clean numerical simulation (CNS) for chaos, Adv. Appl. Math. Mech. 15 (5) (2023) 1191–1215, https://doi.org/10.4208/aamm.OA-2022-0340.
[39] J.A. Pérez-Hernández, L. Benet, PerezHz/TaylorIntegration.jl: TaylorIntegration, https://doi.org/10.5281/zenodo.2562353, Feb. 2019.
[40] A. Abad, R. Barrio, F. Blesa, M. Rodríguez, Algorithm 924: TIDES, a Taylor series integrator for differential equations, ACM Trans. Math. Softw. 39 (1) (2012) 1–28.
[41] R. Barrio, F. Blesa, M. Lara, VSVO formulation of the Taylor method for the numerical solution of ODEs, Comput. Math. Appl. 50 (1–2) (2005) 93–111.
[42] A. Meurer, C.P. Smith, M. Paprocki, O. Čertík, S.B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J.K. Moore, S. Singh, T. Rathnayake, S. Vig, B.E. Granger, R.P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M.J. Curry, A.R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, A. Scopatz, Sympy: symbolic computing in Python, PeerJ Comput. Sci. 3 (2017) e103, https://doi.org/10.7717/peerj-cs.103.
[43] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, P. Zimmermann, Mpfr: a multiple-precision binary floating-point library with correct rounding, ACM Trans. Math. Softw. 33 (2) (2007) 13.
[44] C. Simó, Global dynamics and fast indicators, in: Global Analysis of Dynamical Systems, CRC Press, 2001, pp. 376–392.