Forward and adjoint sensitivity computation of chaotic dynamical systems

Forward and Adjoint Sensitivity Computation of Chaotic Dynamical Systems

Qiqi Wang [a,*],

[a] *Department of Aeronautics and Astronautics, MIT, 77 Mass Ave, Cambridge, MA 02139, USA*

Abstract

This paper describes a forward algorithm and an adjoint algorithm for computing sensitivity derivatives in chaotic dynamical systems, such as the Lorenz attractor. The algorithms compute the derivative of long time averaged "statistical" quantities to infinitesimal perturbations of the system parameters. The algorithms are demonstrated on the Lorenz attractor. We show that sensitivity derivatives of statistical quantities can be accurately estimated using a single, short trajectory (over a time interval of 20) on the Lorenz attractor.

Key words: Sensitivity analysis, linear response, adjoint equation, unsteady adjoint, chaos, statistical average, Lyapunov exponent, Lyapunov covariant vector, Lorenz attractor.

1 Introduction

Computational methods for sensitivity analysis is a powerful tool in modern computational science and engineering. These methods calculate the derivatives of output quantities with respect to input parameters in computational simulations. There are two types of algorithms for computing sensitivity derivatives: the forward algorithms and the adjoint algorithms. The forward algorithms are more efficient for computing sensitivity derivatives of many output quantities to a few input parameters; the adjoint algorithms are more efficient for computing sensitivity derivatives of a few output quantities to many input parameters. Key application of computational methods for sensitivity analysis

* Corresponding author.
Email address: qiqi@mit.edu (Qiqi Wang).

include aerodynamic shape optimization [3], adaptive grid refinement [9], and data assimilation for weather forecasting [8].

In simulations of chaotic dynamical systems, such as turbulent flows and the climate system, many output quantities of interest are "statistical averages". Denote the state of the dynamical system as $x(t)$; for a function of the state $J(x)$, the corresponding statistical quantity $\langle J \rangle$ is defined as an average of $J(x(t))$ over an infinitely long time interval:

$$\langle J \rangle = \lim_{T \to \infty} \frac{1}{T} \int_0^T J(x(t))\, dt \;, \tag{1}$$

For ergodic dynamical systems, a statistical average only depends on the governing dynamical system, and does not depend on the particular choice of trajectory $x(t)$.

Many statistical averages, such as the mean aerodynamic forces in turbulent flow simulations, and the mean global temperature in climate simulations, are of great scientific and engineering interest. Computing sensitivities of these statistical quantities to input parameters can be useful in many applications.

The differentiability of these statistical averages to parameters of interest as been established through the recent developments in the Linear Response Theory for dissipative chaos [6][7]. A class of chaotic dynamical systems, known as "quasi-hyperbolic" systems, has been proven to have statistical quantities that are differentiable with respect to small perturbations. These systems include the Lorenz attractor, and possibly many systems of engineering interest, such as turbulent flows.

Despite recent advances both in Linear Response Theory [7] and in numerical methods for sensitivity computation of unsteady systems [10] [4], sensitivity computation of statistical quantities in chaotic dynamical systems remains difficult. A major challenge in computing sensitivities in chaotic dynamical systems is their sensitivity to the initial condition, commonly known as the "butterfly effect". The linearized equations, used both in forward and adjoint sensitivity computations, give rise to solutions that blow up exponentially. When a statistical quantity is approximated by a finite time average, the computed sensitivity derivative of the finite time average diverges to infinity, instead of converging to the sensitivity derivative of the statistical quantity [5]. Existing methods for computing correct sensitivity derivatives of statistical quantities usually involve averaging over a large number of ensemble calculations [5] [1]. The resulting high computation cost makes these methods not attractive in many applications.

This paper outlines a computational method for efficiently estimating the sensitivity derivative of time averaged statistical quantities, relying on a single

trajectory over a small time interval. The key idea of our method, inversion of the "shadow" operator, is already used as a tool for proving structural stability of strange attractors [6]. The key strategy of our method, divide and conquer of the shadow operator, is inspired by recent advances in numerical computation of the Lyapunov covariant vectors [2][11].

In the rest of this paper, Section 2 describes the "shadow" operator, on which our method is based. Section 3 derives the sensitivity analysis algorithm by inverting the shadow operator. Section 4 introduces a fix to the singularity of the shadow operator. Section 5 summarizes the forward sensitivity analysis algorithm. Section 6 derives the corresponding adjoint version of the sensitivity analysis algorithm. Section 7 demonstrates both the forward and adjoint algorithms on the Lorenz attractor. Section 8 concludes this paper.

The paper uses the following mathematical notation: Vector fields in the state space (e.g. $f(x)$, $\phi_i(x)$) are column vectors; gradient of scalar fields (e.g. $\frac{\partial a_i^x}{\partial x}$) are row vectors; gradient of vector fields (e.g. $\frac{\partial f}{\partial x}$) are matrices with each row being a dimension of f, and each column being a dimension of x. The $(\cdot)$ sign is used to identify matrix-vector products or vector-vector inner products. For a trajectory $x(t)$ satisfying $\frac{dx}{dt} = f(x)$ and a scalar or vector field $a(x)$ in the state space, we often use $\frac{da}{dt}$ to denote $\frac{da(x(t))}{dt}$. The chain rule $\frac{da}{dt} = \frac{da}{dx} \cdot \frac{dx}{dt} = \frac{da}{dx} \cdot f$ is often used without explanation.

2 The "Shadow Operator"

For a smooth, uniformly bounded n dimensional vector field $\delta x(x)$, defined on the n dimensional state space of x. The following transform defines a slightly "distorted" coordinates of the state space:

$$x'(x) = x + \epsilon\, \delta x(x) \tag{2}$$

where ϵ is a small real number. Note that for an infinitesimal ϵ, the following relation holds:

$$x'(x) - x = \epsilon\, \delta x(x) = \epsilon\, \delta x(x') + O(\epsilon^2) \tag{3}$$

We call the transform from x to x' as a "shadow coordinate transform". In particular, consider a trajectory $x(t)$ and the corresponding transformed trajectory $x'(t) = x'(x(t))$. For a small ϵ, the transformed trajectory $x'(t)$ would "shadow" the original trajectory $x(t)$, i.e., it stays uniformly close to $x(t)$ forever. Figure 1 shows an example of a trajectory and its shadow.

Now consider a trajectory $x(t)$ satisfying an ordinary differential equation

$$\dot{x} = f(x)\ , \tag{4}$$

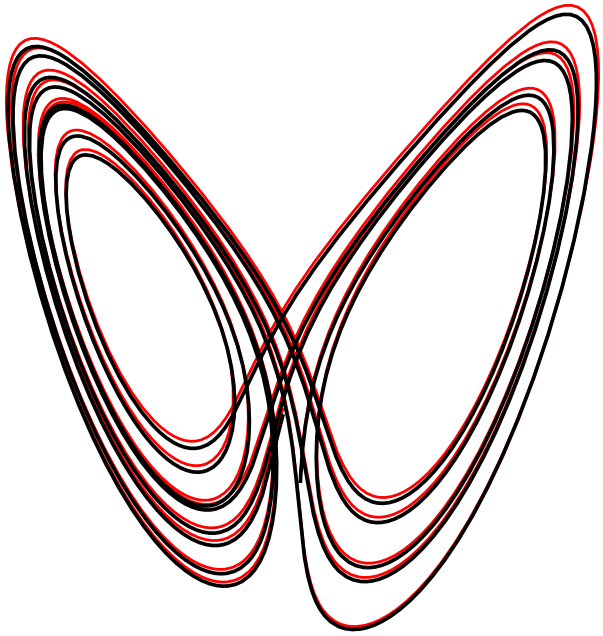


Fig. 1. A trajectory of the Lorenz attractor under a shadow coordinate transform. The black trajectory shows $x(t)$, and the red trajectory shows $x'(t)$. The perturbation $\epsilon\,\delta x$ shown corresponds to an infinitesimal change in the parameter r, and is explained in detail in Section 7.

with a smooth vector field $f(x)$ as a function of x. The same trajectory in the transformed "shadow" coordinates $x'(t)$ do not satisfy the same differential equation. Instead, from Equation (3), we obtain

$$\begin{aligned} \dot{x'} &= f(x) + \epsilon \frac{\partial \delta x}{\partial x} \cdot f(x) \\ &= f(x') - \epsilon \frac{\partial f}{\partial x} \cdot \delta x(x') + \epsilon \frac{\partial \delta x}{\partial x} \cdot f(x') + O(\epsilon^2) \end{aligned} \tag{5}$$

In other words, the shadow trajectory $x'(t)$ satisfies a slightly perturbed equation

$$\dot{x'} = f(x') + \epsilon\,\delta f(x') + O(\epsilon^2) \tag{6}$$

where the perturbation δf is

$$\begin{aligned} \delta f(x) &= -\frac{\partial f}{\partial x} \cdot \delta x(x) + \frac{\partial \delta x}{\partial x} \cdot f(x) \\ &= -\frac{\partial f}{\partial x} \cdot \delta x(x) + \frac{d\delta x}{dt} \\ &:= (S_f \delta x)(x) \end{aligned} \tag{7}$$

For a given differential equation $\dot{x} = f(x)$, Equation (7) defines a linear operator $S_f : \delta x \Rightarrow \delta f$. We call S_f the "**Shadow Operator**" of f. For any smooth vector field $\delta x(x)$ that defines a slightly distorted "shadow" coordinate system in the state space, S_f determines a unique smooth vector field

$\delta f(x)$ that defines a perturbation to the differential equation. Any trajectory of the original differential equation would satisfy the perturbed equation in the distorted coordinates.

Given an ergodic dynamical system $\dot{x} = f(x)$, and a pair $(\delta x, \delta f)$ that satisfies $\delta f = S_f \delta x$, δx determines the **sensitivity of statistical quantities** of the dynamical system to an infinitesimal perturbation $\epsilon \delta f$. Let $J(x)$ be a smooth scalar function of the state, consider the statistical average $\langle J \rangle$ as defined in Equation (1). The sensitivity derivative of $\langle J \rangle$ to the infinitesimal perturbation $\epsilon\, \delta f$ is by definition

$$\frac{d\langle J \rangle}{d\epsilon} = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \left(\lim_{T \to \infty} \frac{1}{T} \int_0^T J(x'(t))\, dt - \lim_{T \to \infty} \frac{1}{T} \int_0^T J(x(t))\, dt \right) \tag{8}$$

where by the ergodicity assumption, the statistical average of the perturbed system can be computed by averaging over $x'(t)$, which satisfies the perturbed governing differential equation. Continuing from Equation (8),

$$\begin{aligned} \frac{d\langle J \rangle}{d\epsilon} &= \lim_{\epsilon \to 0} \lim_{T \to \infty} \frac{1}{T} \int_0^T \frac{J(x'(t)) - J(x(t))}{\epsilon}\, dt \\ &= \lim_{T \to \infty} \lim_{\epsilon \to 0} \frac{1}{T} \int_0^T \frac{J(x'(t)) - J(x(t))}{\epsilon}\, dt \\ &= \lim_{T \to \infty} \frac{1}{T} \int_0^T \frac{\partial J}{\partial x} \cdot \delta x\, dt = \left\langle \frac{\partial J}{\partial x} \cdot \delta x \right\rangle . \end{aligned} \tag{9}$$

Equation (9) represents the sensitivity derivative of a statistical quantity $\langle J \rangle$ to the size of a perturbation $\epsilon \delta f$. There are two subtle points here:

- The two limits $\lim_{\epsilon \to 0}$ and $\lim_{T \to \infty}$ can commute with each other for the following reason: The two trajectories $x'(t)$ and $x(t)$ stay **uniformly** close to each other **forever**; therefore,

$$\frac{J(x'(t)) - J(x(t))}{\epsilon} \xrightarrow{\epsilon \to 0} \frac{\partial J}{\partial x} \cdot \delta x \tag{10}$$

  uniformly for all t. Consequently,

$$\frac{1}{T} \int_0^T \frac{J(x'(t)) - J(x(t))}{\epsilon}\, dt \xrightarrow{\epsilon \to 0} \frac{1}{T} \int_0^T \frac{\partial J}{\partial x} \cdot \delta x\, dt \tag{11}$$

  uniformly for all T. Thus the two limits commute.
- The two trajectories $x'(t)$ and $x(t)$ start at two specially positioned pair of initial conditions $x'(0) = x(0) + \epsilon\, \delta x(x(0))$. Almost any other pair of initial conditions (e.g. $x'(0) = x(0)$) would make the two trajectories diverge as a result of the "butterfly effect". They would not stay uniformly close to each other, and the limits $\lim_{\epsilon \to 0}$ and $\lim_{T \to \infty}$ would not commute.

Equation (9) represents the sensitivity derivative of the statistical quantity $\langle J \rangle$ to the infinitesimal perturbation $\epsilon\, \delta f$ as another statistical quantity $\langle \frac{\partial J}{\partial x} \cdot \delta x \rangle$. We can compute it by averaging $\frac{\partial J}{\partial x} \cdot \delta x$ over a sufficiently long trajectory, provided that $\delta x = S^{-1} \delta f$ is known along the trajectory. The next section describes how to numerically compute $\delta x = S^{-1} \delta f$ for a given δf.

3 Inverting the Shadow Operator

Perturbations to input parameters can often be represented as perturbations to the dynamics. Consider a differential equation $\dot{x} = f(x, s_1, s_2, \ldots, s_m)$ parameterized by m input variables, an infinitesimal perturbation in a input parameter $s_j \to s_j + \epsilon$ can be represented as a perturbation to the dynamics $\epsilon\, \delta f = \epsilon \frac{df}{ds_j}$.

Equation (9) defines the sensitivity derivative of the statistical quantity $\langle J \rangle$ to an infinitesimal perturbation $\epsilon\, \delta f$, provided that a δx can be found satisfying $\delta f = S_f \delta x$, where S_f is the shadow operator. To compute the sensitivity by evaluating Equation (9), one must first numerically invert S_f for a given δf to find the corresponding δx.

The key ingredient of numerical inversion of S_f is the Lyapunov spectrum decomposition. This decomposition can be efficiently computed numerically [11] [2]. In particular, we focus on the case when the system $\dot{x} = f(x)$ has distinct Lyapunov exponents. Denote the Lyapunov covariant vectors as $\phi_1(x), \phi_2(x), \ldots, \phi_n(x)$. Each ϕ_i is a vector field in the state space satisfying

$$\frac{d}{dt}\phi_i(x(t)) = \frac{\partial f}{\partial x} \cdot \phi_i(x(t)) - \lambda_i \phi_i(x(t)) \tag{12}$$

where $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the Lyapunov exponents in decreasing order.

The Lyapunov spectrum decomposition enables a divide and conquer strategy for computing $\delta x = S_f^{-1} \delta f$. For any $\delta f(x)$ and every point x on the attractor, both $\delta x(x)$ and $\delta f(x)$ can be decomposed into the Lyapunov covariant vector directions almost everywhere, i.e.

$$\delta x(x) = \sum_{i=1}^{n} a_i^x(x)\, \phi_i(x) \ , \tag{13}$$

$$\delta f(x) = \sum_{i=1}^{n} a_i^f(x)\, \phi_i(x) \ , \tag{14}$$

where a_i^x and a_i^f are scalar fields in the state space. From the form of S_f in

Equation (7), we obtain

$$\begin{aligned} S_f(a_i^x \phi_i) &= -\frac{\partial f}{\partial x} \cdot (a_i^x(x)\phi_i(x)) + \frac{d}{dt}(a_i^x(x)\,\phi_i(x)) \\ &= -\,a_i^x(x)\,\frac{\partial f}{\partial x} \cdot \phi_i(x) + \frac{d\,a_i^x(x)}{dt}\,\phi_i(x) + a_i^x(x)\,\frac{d\,\phi_i(x)}{dt}\ . \end{aligned} \tag{15}$$

By substituting Equation (12) into the last term of Equation (15), we obtain

$$S_f(a_i^x \phi_i) = \left(\frac{da_i^x(x)}{dt} - \lambda_i\, a_i^x(x)\right)\,\phi_i(x)\ , \tag{16}$$

By combining Equation (16) with Equations (13), (14) and the linear relation $\delta f = S_f \delta x$, we finally obtain

$$\delta f = \sum_{i=1}^{n} S_f(a_i^x \phi_i) = \sum_{i=1}^{n} \underbrace{\left(\frac{da_i^x}{dt} - \lambda_i\, a_i^x\right)}_{a_i^f}\,\phi_i\ , \tag{17}$$

Equations (16) and (17) are useful for two reasons:

(1) They indicate that the Shadow Operator S_f, applied to a scalar field $a_i^x(x)$ multiple of $\phi_i(x)$, generates another scalar field $a_i^f(x)$ multiple of the same vector field $\phi_i(x)$. Therefore, one can compute $S_f^{-1}\delta f$ by first decomposing δf as in Equation (14) to obtain the a_i^f. If each a_i^x can be calculated from the corresponding a_i^f, then δx can be computed with Equation (13), completing the inversion.

(2) It defines a scalar ordinary differential equation that governs the relation between a_i^x and a_i^f along a trajectory $x(t)$:

$$\frac{da_i^x(x)}{dt} = a_i^f(x) + \lambda_i\, a_i^x(x) \tag{18}$$

This equation can be used to obtain a_i^x from a_i^f along a trajectory, thereby filling the gap in the inversion procedure of S_f outlined above. For each positive Lyapunov exponent λ_i, one can integrate the ordinary differential equation

$$\frac{d\check{a}_i^x}{dt} = \check{a}_i^f + \lambda_i\, \check{a}_i^x \tag{19}$$

backwards in time from an arbitrary terminal condition, and the difference between $\check{a}_i^x(t)$ and the desired $a_i^x(x)$ will decrease exponentially. For each negative Lyapunov exponent λ_i, Equation (19) can be integrated forward in time from an arbitrary initial condition, and $\check{a}_i^x(t)$ will converge exponentially to the desired $a_i^x(x)$. For a zero Lyapunov exponent $\lambda_i = 0$, Section 4 introduces a solution.

4 Time Dilation and Compression

There is a fundamental problem in the inversion method derived in Section 3: S_f is not invertible for certain δf. This can be shown with the following analysis: Any continuous time dynamical system with a non-trivial attractor must have a zero Lyapunov exponent $\lambda_{n_0} = 0$. The corresponding Lyapunov covariant vector is $\phi_{n_0}(x) = f(x)$. This can be verified by substituting $\lambda_i = 0$ and $\phi_i = f$ into Equation (12). For this $i = n_0$, Equations (19) becomes

$$a^f_{n_0}(x) = \frac{da^x_{n_0}(x)}{dt} \tag{20}$$

By taking an infinitely long time average on both sides of Equation (20), we obtain

$$\left\langle a^f_{n_0}(x) \right\rangle = \lim_{T\to\infty} \frac{a^x_{n_0}(x(T)) - a^x_{n_0}(x(0))}{T} = 0 \,, \tag{21}$$

Equation (21) implies that for any $\delta f = S_f \delta x$, the $i = n_0$ component of its Lyapunov decomposition (as in Equation (14)) must satisfy $\langle a^f_{n_0}(x)\rangle = 0$. Any δf that do not satisfy this linear relation, e.g. $\delta f \equiv f$, would not be in the range space of S_f. Thus the corresponding $\delta x = S_f^{-1}\delta f$ does not exist.

Our solution to the problem is complementing S_f with a "global time dilation and compression" constant η, whose effect produces a δf that is outside the range space of S_f. We call η a time dilation constant for short. The combined effect of a time dilation constant and a shadow transform could produce all smooth perturbations δf.

The idea comes from the fact that for a constant η, the time dilated or compressed system $\dot{x} = (1+\epsilon\,\eta) f(x)$ has exactly the same statistics $\langle J\rangle$, as defined in Equation (1), as the original system $\dot{x} = f(x)$. Therefore, the perturbation in any $\langle J\rangle$ due to any $\epsilon\,\delta f$ is equal to the perturbation in $\langle J\rangle$ due to $\epsilon\,(\eta f(x) + \delta f(x))$. Therefore, the sensitivity derivative to δf can be computed if we can find a δx that satisfies $S_f \delta x = \eta f(x) + \delta f(x)$ for some η.

We use the "free" constant η to put $\eta f(x) + \delta f(x)$ into the range space of S_f. By substituting $\eta f(x) + \delta f(x)$ into the constraint Equation (21) that identifies the range space of S_f, the appropriate η must satisfy the following equation

$$\eta + \langle a^f_{n_0}\rangle = 0 \,, \tag{22}$$

which we use to numerically compute η.

Once the appropriate time dilation constant η is computed, $\eta f(x) + \delta f(x)$ is in the range space of S_f. We use the procedure in Section 3 to compute $\delta x = S_f^{-1}(\eta f + \delta f)$, then use Equation (9) to compute the desired sensitivity

derivative $d\langle J\rangle/d\epsilon$. The addition of ηf to δf affects Equation (19) only for $i = n_0$, making it

$$\frac{da^x_{n_0}(x)}{dt} = a^f_{n_0}(x) + \eta \; . \tag{23}$$

Equation (23) indicates that $a^x_{n_0}$ can be computed by integrating the right hand side along the trajectory.

The solution to Equation (23) admits an arbitrary additive constant. The effect of this arbitrary constant is the following: By substituting Equations (13) into Equation (9), the contribution from the $i = n_0$ term of δx to $d\langle J\rangle/d\epsilon$ is

$$\lim_{T\to\infty} \frac{1}{T} \int_0^T a^f_{n_0} \frac{dJ}{dt}\, dt \tag{24}$$

Therefore, any constant addition to $a^f_{n_0}$ vanishes as $T \to \infty$. Computationally, however, Equation (9) must be approximated by a finite time average. We find it beneficial to adjust the level of $a^f_{n_0}$ to approximately $\langle a^f_{n_0}\rangle = 0$, in order to control the error due to finite time averaging.

5 The Forward Sensitivity Analysis Algorithms

For a given $\dot{x} = f(x)$, δf and $J(x)$, the mathematical developments in Sections 3 and 4 are summarized into Algorithm 1 for computing the sensitivity derivative $d\delta\langle J\rangle/d\epsilon$ as in Equation (9).

The preparation phase of the algorithm (Steps 1-3) computes a trajectory and the Lyapunov spectrum decomposition along the trajectory. The algorithm then starts by decomposing δf (Step 4), followed by computing δx (Steps 5-7), and finally computing $d\langle J\rangle/d\epsilon$ (Step 8). The sensitivity derivative of many different statistical quantities $\langle J_1\rangle, \langle J_2\rangle, \ldots$ to a single δf can be computed by only repeating the last step of the algorithm. Therefore, this is a "forward" algorithm in the sense that it efficiently computes sensitivity of multiple output quantities to a single input parameter (the size of perturbation $\epsilon\,\delta f$). We will see that this is in sharp contrast to the "adjoint" algorithm described in Section 6, which efficiently computes the sensitivity derivative of one output statistical quantity $\langle J\rangle$ to many perturbations $\delta f_1, \delta f_2, \ldots$.

It is worth noting that the δx computed using Algorithm 1 satisfies the forward tangent equation

$$\dot{\delta x} = \frac{\partial f}{\partial x} \cdot \delta x + \eta\, f + \delta f \tag{25}$$

This can be verified by taking derivative of Equation (13), substituting Equations (19) and (23), then using Equation (14). However, δx must satisfy both an initial condition and a terminal condition, making it difficult to solve with conventional time integration methods. In fact, Algorithm 1 is equivalent to

Algorithm 1 The Forward Sensitivity Analysis Algorithm

(1) Choose a "spin-up buffer time" T_B, and an "statistical averaging time" T_A. T_B should be much longer than $1/|\lambda_i|$ for all nonzero Lyapunov exponent λ_i, so that the solutions of Equation (19) can reach a_i^x over a time span of T_B. T_A should be much longer than the decorrelation time of the dynamics, so that one can accurately approximate a statistical quantity by averaging over $[0, T_A]$.
(2) Obtain an initial condition on the attractor at $t = -T_B$, e.g., by solving $\dot{x} = f(x)$ for a sufficiently long time span, starting from an arbitrary initial condition.
(3) Solve $\dot{x} = f(x)$ to obtain a trajectory $x(t), t \in [-T_B, T_A + T_B]$; compute the Lyapunov exponents λ_i and the Lyapunov covariant vectors $\phi_i(x(t))$ along the trajectory, e.g., using algorithms in [11] and [2].
(4) Perform the Lyapunov spectrum decomposition of $\delta f(x)$ along the trajectory $x(t)$ to obtain $a_i^f(x), i = 1, \ldots, n$ as in Equation (14).
(5) Compute the global time dilation constant η using Equation (22).
(6) Solve the differential equations (19) to obtain a_i^x over the time interval $[0, T_A]$. The equations with positive λ_i are solved backward in time from $t = T_A + T_B$ to $t = 0$; the ones with negative λ_i are solved forward in time from $t = -T_B$ to $t = T_A$. For $\lambda_{n_0} = 0$, Equation (23) is integrated, and the mean of $a_{n_0}^x$ is set to zero.
(7) Compute δx along the trajectory $x(t), t \in [0, T_A]$ with Equation (13).
(8) Compute $d\langle J \rangle / d\epsilon$ using Equation (1) by averaging over the time interval $[0, T_A]$.

splitting δx into stable, neutral and unstable components, corresponding to positive, zero and negative Lyapunov exponents; then solving Equation (25) separately for each component in different time directions. This alternative version of the forward sensitivity computation algorithm could be useful for large systems to avoid computation of all the Lyapunov covariant vectors.

6 The Adjoint Sensitivity Analysis Algorithm

The adjoint algorithm starts by trying to find an **adjoint vector field** $\hat{f}(x)$, such that the sensitivity derivative of the given statistical quantity $\langle J \rangle$ to any infinitesimal perturbation $\epsilon\, \delta f$ can be represented as

$$\frac{d\langle J \rangle}{\epsilon} = \left\langle \hat{f}^{\,T} \cdot \delta f \right\rangle \tag{26}$$

Both $\hat{f}$ in Equation (26) and $\frac{\partial J}{\partial x}$ in Equation (9) can be decomposed into linear combinations of the *adjoint Lyapunov covariant vectors* almost everywhere on

the attractor:

$$\hat{f}(x) = \sum_{i=1}^{n} \hat{a}_i^f(x)\, \psi_i(x) \ , \tag{27}$$

$$\frac{\partial J}{\partial x}^T = \sum_{i=1}^{n} \hat{a}_i^x(x)\, \psi_i(x) \ , \tag{28}$$

where the adjoint Lyapunov covariant vectors ψ_i satisfy

$$-\frac{d}{dt}\psi_i(x(t)) = \frac{\partial f}{\partial x}^T \cdot \psi_i(x(t)) - \lambda_i \psi_i(x(t)) \tag{29}$$

With proper normalization, the (primal) Lyapunov covariant vectors ϕ_i and the adjoint Lyapunov covariant vectors ψ_i have the following conjugate relation:

$$\psi_i(x)^T \cdot \phi_j(x) \equiv \begin{cases} 0 \ , & i \neq j \\ 1 \ , & i = j \end{cases} \tag{30}$$

i.e., the $n \times n$ matrix formed by all the ϕ_i and the $n \times n$ matrix formed by all the ψ_i are the transposed inverse of each other at every point x in the state space.

By substituting Equations (13) and (28) into Equation (9), and using the conjugate relation in Equation (30), we obtain

$$\frac{d\langle J \rangle}{d\epsilon} = \sum_{i=1}^{n} \langle \hat{a}_i^x a_i^x \rangle \tag{31}$$

Similarly, by combining Equations (26), (14), (27) and (30), it can be shown that $\hat{f}$ satisfies Equation (26) if and only if

$$\frac{d\langle J \rangle}{d\epsilon} = \sum_{i=1}^{n} \left\langle \hat{a}_i^f a_i^f \right\rangle \tag{32}$$

Comparing Equations (31) and (32) leads to the following conclusion: Equation (26) can be satisfied by finding $\hat{a}_i^f$ that satisfy

$$\left\langle \hat{a}_i^f a_i^f \right\rangle = \langle \hat{a}_i^x a_i^x \rangle \ , \quad i = 1, \ldots, n \tag{33}$$

The $\hat{a}_i^f$ that satisfies Equation (33) can be found using the relation between a_i^f and a_i^x in Equation (18). By multiplying $\hat{a}_i^f$ on both sides of Equation (18) and integrate by parts in time, we obtain

$$\frac{1}{T}\int_0^T \hat{a}_i^f a_i^f \, dt = \left. \frac{\hat{a}_i^f a_i^x}{T} \right|_0^T - \frac{1}{T}\int_0^T \left(\frac{d\hat{a}_i^f}{dt} + \lambda_i \, \hat{a}_i^f \right) a_i^x \ dt \tag{34}$$

for $i \neq n_0$. Through apply the same technique to Equation (23), we obtain for $i = n_0$

$$\frac{1}{T}\int_0^T \hat{a}^f_{n_0} a^f_{n_0} dt = \left.\frac{\hat{a}^f_{n_0} a^x_{n_0}}{T}\right|_0^T - \frac{1}{T}\int_0^T \frac{d\hat{a}^f_{n_0}}{dt}\, a^x_{n_0} dt + \frac{1}{T}\int_0^T \eta\, \hat{a}^f_{n_0} dt \tag{35}$$

If we set $\hat{a}^f_i$ to satisfy the following relations

$$\begin{aligned} -\frac{d\hat{a}^f_i(x)}{dt} &= \hat{a}^x_i(x) + \lambda_i\, \hat{a}^f_i(x)\ , \qquad && i \neq n_0\ , \\ -\frac{d\hat{a}^f_i(x)}{dt} &= \hat{a}^x_i(x)\ , \quad \langle \hat{a}^f_i \rangle = 0\ , && i = n_0\ , \end{aligned} \tag{36}$$

then Equations (34) and (35) become

$$\begin{aligned} &\frac{1}{T}\int_0^T \hat{a}^f_i a^f_i dt = \left.\frac{\hat{a}^f_i a^x_i}{T}\right|_0^T + \frac{1}{T}\int_0^T \hat{a}^x_i a^x_i\ dt\ , && i \neq n_0 \\ &\frac{1}{T}\int_0^T \hat{a}^f_i a^f_i dt = \left.\frac{\hat{a}^f_i a^x_i}{T}\right|_0^T + \frac{1}{T}\int_0^T \hat{a}^x_i a^x_i\ dt + \eta\left(\frac{1}{T}\int_0^T \hat{a}^f_{n_0}\, dt - \langle \hat{a}^f_{n_0} \rangle\right), && i = n_0 \end{aligned} \tag{37}$$

As $T \to \infty$, both equations reduces to Equation (33).

In summary, if the scalar fields $\hat{a}^f_i$ satisfy Equation (36), then they also satisfy Equation (37) and thus Equation (33); as a result, the $\hat{f}$ formed by these $\hat{a}^f$ through Equation (27) satisfies Equation (26), thus is the desired adjoint vector field.

For each $i \neq n_0$, the scalar field $\hat{a}^f_i$ satisfying Equation (36) can be computed by solving an ordinary differential equations

$$-\frac{d\check{\hat{a}}^f_i}{dt} = \check{\hat{a}}^x_i + \lambda_i\, \check{\hat{a}}^f_i\ . \tag{38}$$

Contrary to computation of a^x_i through solving Equation (19), the time integration should be forward in time for positive λ_i, and backward in time for negative λ_i, in order for the difference between $\check{\hat{a}}^f_i(t)$ and $\hat{a}^f_i(x(t))$ to diminish exponentially.

The $i = n_0$ equation in Equation (36) can be directly integrated to obtain $\hat{a}^f_{n_0}(x)$. The equation is well defined because the right hand side is mean zero:

$$\frac{1}{T}\int_0^T \hat{a}^f_{n_0}(x(t))\, dt = \frac{1}{T}\int_0^T \frac{\partial J}{\partial x} \cdot \phi_{n_0}\, dt = \frac{1}{T}\int_0^T \frac{dJ}{dt}\, dt \overset{T\to\infty}{\longrightarrow} 0\ . \tag{39}$$

Therefore, the integral of $\hat{a}^x_{n_0}(x)$ over time, subtracted by its mean, is the solution $\hat{a}^f_{n_0}(x)$ to the $i = n_0$ case of Equation (36).

Algorithm 2 The Adjoint Sensitivity Analysis Algorithm

(1) Choose a "spin-up buffer time" T_B, and an "statistical averaging time" T_A. T_B should be much longer than $1/|\lambda_i|$ for all nonzero Lyapunov exponent λ_i, so that the solutions of Equation (19) can reach a_i^x over a time span of T_B. T_A should be much longer than the decorrelation time of the dynamics, so that one can accurately approximate a statistical quantity by averaging over $[0, T_A]$.

(2) Obtain an initial condition on the attractor at $t = -T_B$, e.g., by solving $\dot{x} = f(x)$ for a sufficiently long time span, starting from an arbitrary initial condition.

(3) Solve $\dot{x} = f(x)$ to obtain a trajectory $x(t), t \in [-T_B, T_A + T_B]$; compute the Lyapunov exponents λ_i and the Lyapunov covariant vectors $\phi_i(x(t))$ along the trajectory, e.g., using algorithms in [11] and [2].

(4) Perform the Lyapunov spectrum decomposition of $(\partial J/\partial x)^T$ along the trajectory $x(t)$ to obtain $\hat{a}_i^x(x(t)), i = 1, \ldots, n$ as in Equation (28).

(5) Solve the differential equations (38) to obtain $\hat{a}_i^f(x(t))$ over the time interval $[0, T_A]$. The equations with negative λ_i are solved backward in time from $t = T_A + T_B$ to $t = 0$; the ones with positive λ_i are solved forward in time from $t = -T_B$ to $t = T_A$. For $i = n_0$, the scalar $-a_{n_0}^x$ is integrated along the trajectory; the mean of the integral is subtracted from the integral itself to obtain $\hat{a}_{n_0}^f$.

(6) Compute $\hat{f}$ along the trajectory $x(t), t \in [0, T_A]$ with Equation (27).

(7) Compute $d\langle J\rangle/d\epsilon$ using Equation (26) by averaging over the time interval $[0, T_A]$.

The above analysis summarizes to Algorithm 2 for computing the sensitivity derivative derivative of the statistical average $\langle J\rangle$ to an infinitesimal perturbations $\epsilon\,\delta f$. The preparation phase of the algorithm (Steps 1-3) is exactly the same as in Algorithm 1. These steps compute a trajectory $x(t)$ and the Lyapunov spectrum decomposition along the trajectory. The adjoint algorithm then starts by decomposing the derivative vector $(\partial J/\partial x)^T$ (Step 4), followed by computing the adjoint vector δf (Steps 5-6), and finally computing $d\langle J\rangle/d\epsilon$ for a particular δf. Note that the sensitivity of the same $\langle J\rangle$ to many different perturbations $\delta f_1, \delta f_2, \ldots$ can be computed by repeating only the last step of the algorithm. Therefore, this is an "adjoint" algorithm, in the sense that it efficiently computes the sensitivity derivatives of a single output quantity to many input perturbation.

It is worth noting that $\hat{f}$ computed using Algorithm 2 satisfies the adjoint equation

$$-\dot{\hat{f}} = \frac{\partial f}{\partial x}^T \cdot \hat{f} - \frac{\partial J}{\partial x} \tag{40}$$

This can be verified by taking derivative of Equation (27), substituting Equation (36), then using Equation (28). However, $\hat{f}$ must satisfy both an initial condition and a terminal condition, making it difficult to solve with conven-

tional time integration methods. In fact, Algorithm 2 is equivalent to splitting $\hat{f}$ into stable, neutral and unstable components, corresponding to positive, zero and negative Lyapunov exponents; then solving Equation (40) separately for each component in different time directions. This alternative version of the adjoint sensitivity computation algorithm could be useful for large systems, to avoid computation of all the Lyapunov covariant vectors.

7 An Example: the Lorenz Attractor

We consider the Lorenz attractor $\dot{x} = f(x)$, where $x = (x_1, x_2, x_3)^T$, and

$$f(x) = \begin{pmatrix} \sigma(x_2 - x_1) \\ x_1(r - x_3) - x_2 \\ x_1 x_2 - \beta x_3 \end{pmatrix} \tag{41}$$

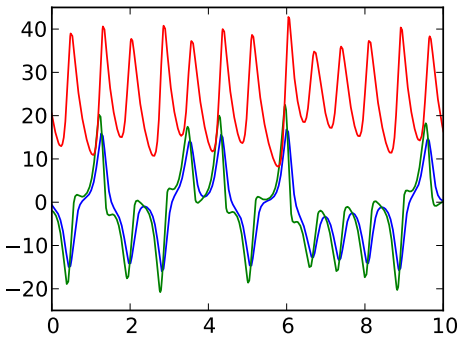
The "classic" parameter values $\sigma = 10$, $r = 28$, $\beta = 8/3$ are used. Both the forward sensitivity analysis algorithm (Algorithm 1) and the adjoint sensitivity analysis algorithm (Algorithm 2) are performed on this system.

We want to demonstrate the computational efficiency of our algorithm; therefore, we choose a relatively short statistical averaging interval of $T_A = 10$, and a spin up buffer period of $T_B = 5$. Only a single trajectory of length $T_A + 2T_B$ on the attractor is required in our algorithm. Considering that the oscillation period of the Lorenz attractor is around 1, the combined trajectory length of 20 is a reasonable time integration length for most simulations of chaotic dynamical systems. In our example, we start the time integration from $t = -10$ at $x = (-8.67139571762, 4.98065219709, 25)$, and integrate the equation to $t = -5$, to ensure that the entire trajectory from $-T_B$ to $T_A + T_B$ is roughly on the attractor. The rest of the discussion in this section are focused on the trajectory $x(t)$ for $t \in [-T_B, T_A + T_B]$.

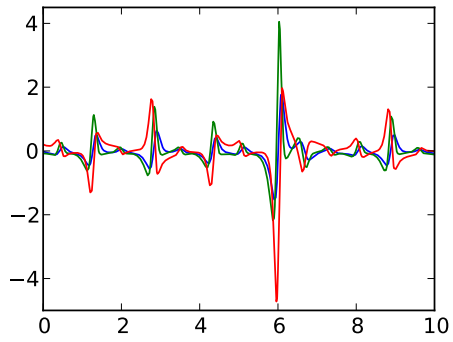
7.1 Lyapunov covariant vectors

The Lyapunov covariant vectors are computed in Step 3 of both Algorithm 1 and Algorithm 2, over the time interval $[-T_B, T_A + T_B]$. These vectors, along with the trajectory $x(t)$, are shown in Figure 2.

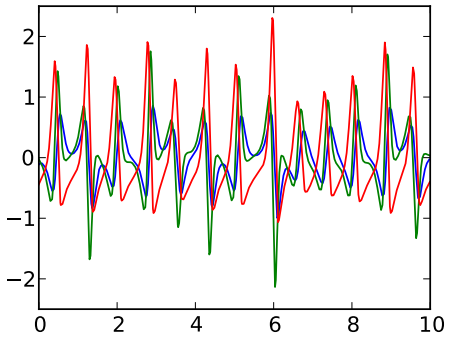
The three dimensional Lorenz attractor has three pairs of Lyapunov exponents and Lyapunov covariant vectors. λ_1 is the only positive Lyapunov exponent,



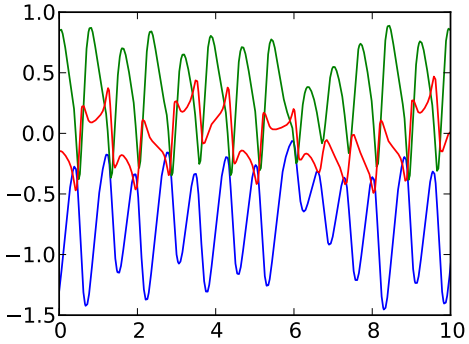
(a) The state vector x



(b) First Lyapunov covariant vector ϕ_1



(c) Second Lyapunov covariant vector ϕ_2



(d) Third Lyapunov covariant vector ϕ_3

Fig. 2. The Lyapunov covariant vectors of the Lorenz attractor along the trajectory $x(t)$ for $t \in [0, 10]$. The x-axes are t; the blue, green and red lines correspond to the x_1, x_2 and x_3 coordinates in the state space, respectively.

and ϕ_1 is computed by integrating the tangent linear equation

$$\dot{\tilde{x}} = \frac{\partial f}{\partial x} \cdot \tilde{x} \tag{42}$$

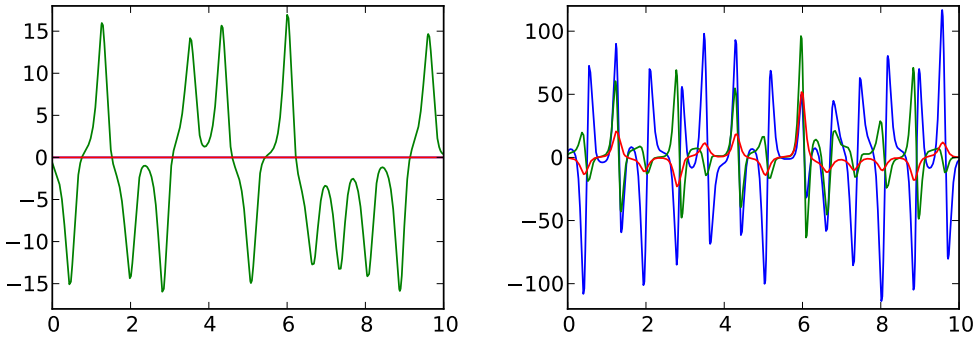
forward in time from an arbitrary initial condition at $t = -T_B$. The first Lyapunov exponent is estimated to be $\lambda_1 \approx 0.95$ through a linear regression of $\tilde{x}$ in the log space. The first Lyapunov vector is then obtained as $\phi_1 = \tilde{x}\, e^{-\lambda_1 t}$.

$\lambda_2 = 0$ is the vanishing Lyapunov exponent; therefore, $\phi_2 = \theta\, f(x)$, where $\theta = 1/\sqrt{\langle \|f\|_2^2 \rangle}$ is a normalizing constant that make the mean magnitude of ϕ_2 equal to 1.

The third Lyapunov exponent λ_3 is negative. So ϕ_3 is computed by integrating the tangent linear equation (42) backwards in time from an arbitrary initial condition at $t = T_A + T_B$. The third Lyapunov exponent is estimated to be $\lambda_3 \approx -14.6$ through a linear regression of the backward solution $\tilde{x}$ in the log space. The third Lyapunov vector is then obtained as $\phi_3 = \tilde{x}\, e^{-\lambda_3 t}$.

We demonstrate our forward sensitivity analysis algorithm by computing the sensitivity derivative of three statistical quantities $\langle x_1^2 \rangle$, $\langle x_2^2 \rangle$ and, $\langle x_3 \rangle$ to a small perturbation in the system parameter r in the Lorenz attractor Equation (41). The infinitesimal perturbation $r \to r + \epsilon$ is equivalent to the perturbation

$$\epsilon \, \delta f = \epsilon \, \frac{\partial f}{\partial r} = \epsilon \, (0, x_1, 0)^T \ . \tag{43}$$



(a) $\delta f = \dfrac{df}{dr}$ (b) $a_i^f, i = 1, 2, 3$ for the δf

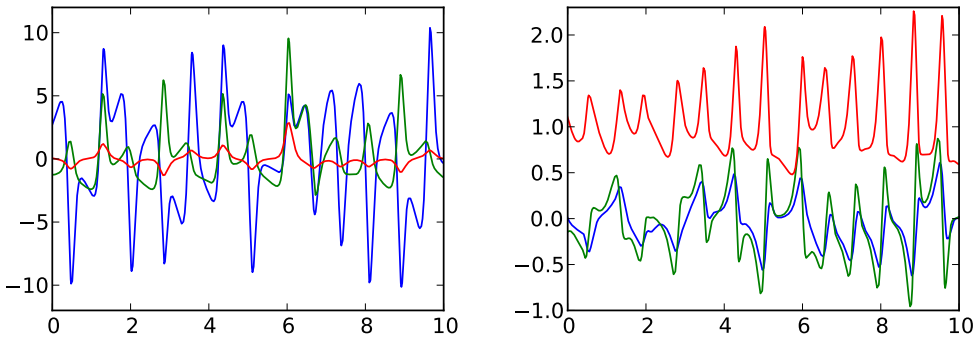
Fig. 3. Lyapunov vector decomposition of δf. The x-axes are t; the blue, green and red lines on the left are the first, second and third component of δf as defined in Equation (43); the blue, green and red lines on the right are a_1^f, a_2^f and a_3^f in the decomposition of δf (Equation (14)), respectively.

The forcing term defined in Equation (43) is plotted in Figure 3a. Figure 3b plots the decomposition coefficients a_i^f, computed by solving a 3×3 linear system defined in Equation (14) at every point on the trajectory.

For each a_i^f obtained through the decomposition, Equation (19) or (23) is solved to obtain a_i^x. For $i = 1$, Equation (19) is solved backwards in time from $t = T_A + T_B$ to $t = 0$. For $i = n_0 = 2$, the time compression constant is estimated to be $\eta \approx -2.78$, and Equation (23) is integrated to obtain a_2^x. For $i = 3$, Equation (19) is solved forward in time from $t = -T_B$ to $t = T_A$.

The resulting values of $a_i^x, i = 1, 2, 3$ are plotted in Figure 4a. These values are then substituted into Equation (13) to obtain δx, as plotted in Figure 4b. The "shadow" trajectory defined as $x' = x + \epsilon \delta x$ is also plotted in Figure 1 as the red lines, for an $\epsilon = 1/3$. This $\delta x = S_f^{-1} \delta f$ is approximately the shadow coordinate perturbation "induced" by a $1/3$ increase in the input parameter r, a.k.a. the Rayleigh number in the Lorenz attractor.

The last step of the forward sensitivity analysis algorithm is computing the



(a) $a_i^x, i = 1, 2, 3$ for the δf (b) $\delta x = \sum_{i=1}^{3} a_i^x \, \phi_i$

Fig. 4. Inversion of S_f for $\delta x = S_f^{-1} \delta f$. The x-axes are t; the blue, green and red lines on the left are a_1^x, a_2^x and a_3^x, respectively; the blue, green and red lines on the right are the first, second and third component of δx, computed via Equation (13).

sensitivity derivatives of the output statistical quantities using Equation (9). We found that using a windowed time averaging [4] yields more accurate sensitivities. Here our estimates over the time interval $[0, T_A]$ are

$$\frac{d\langle x_1^2 \rangle}{dr} \approx 2.64 \, , \quad \frac{d\langle x_2^2 \rangle}{dr} \approx 3.99 \, , \quad \frac{d\langle x_3 \rangle}{dr} \approx 1.01 \tag{44}$$

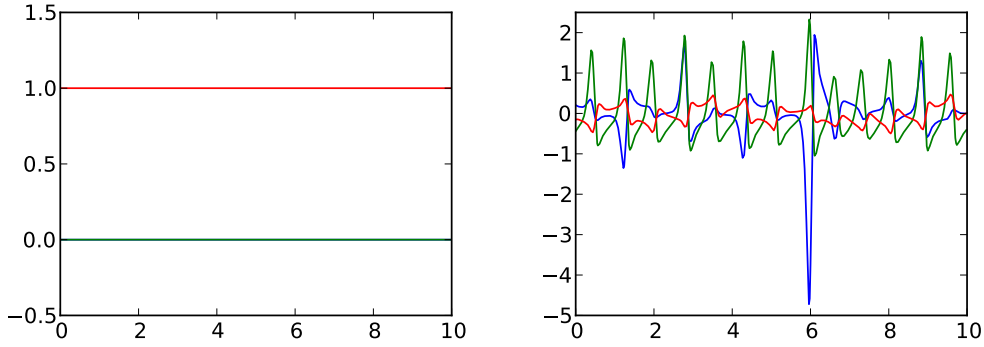
These sensitivity values compare well to results obtained through finite difference, as shown in Section 7.4.

7.3 Adjoint Sensitivity Analysis

We demonstrate our adjoint sensitivity analysis algorithm by computing the sensitivity derivatives of the statistical quantity $\langle x_3 \rangle$ to small perturbations in the three system parameters s, r and b in the Lorenz attractor Equation (41).

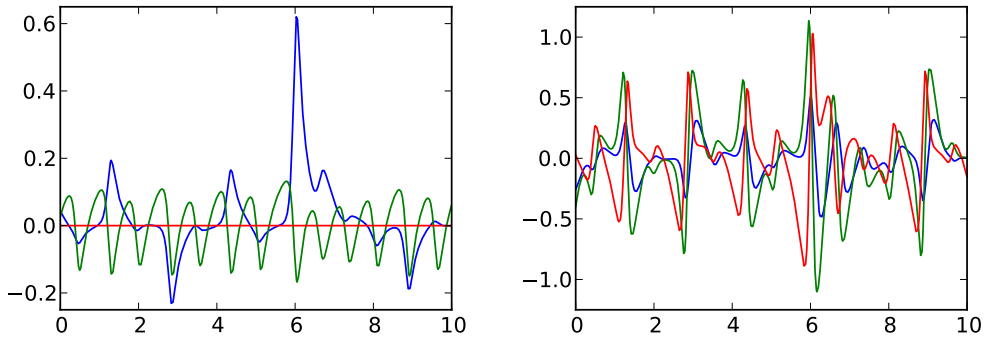
The first three steps of Algorithm 2 is the same as in Algorithm 1, and has been demonstrated in Section 7.1. Step 4 involves decomposing $(\partial J / \partial x)^T$ into three adjoint Lyapunov covariant vectors. In our case, $J(x) = x_3$, therefore $\partial J / \partial x \equiv (0, 0, 1)$, as plotted in Figure 5a. The adjoint Lyapunov covariant vectors ψ_i can be computed using Equation (30) by inverting the 3×3 matrix formed by the (primal) Lyapunov covariant vectors ϕ_i at every point on the trajectory. The coefficients $\hat{a}_i^x, i = 1, 2, 3$ can then be computed by solving Equation (28). These scalar quantities along the trajectory are plotted in Figure 5b for $t \in [0, T_A]$.

Once we obtain $\hat{a}_i^x$, $\hat{a}_i^f$ can be computed by solving Equation (38). The solution is plotted in Figure 6a. Equation (27) can then be used to combine the $\hat{a}_i^f$ into



(a) $\frac{\partial J}{\partial x}$ for $J = x_3$ (b) $\hat{a}_i^x, i = 1, 2, 3$ for the $\frac{\partial J}{\partial x}$

Fig. 5. Adjoint Lyapunov vector decomposition of $\partial J/\partial x$. The x-axes are t; the blue, green and red lines on the left are the first, second and third component of $\partial J/\partial x$; the blue, green and red lines on the right are $\hat{a}_1^x$, $\hat{a}_2^x$ and $\hat{a}_3^x$ in the decomposition of $\partial J/\partial x$ (Equation (28)), respectively.



(a) $\hat{a}_i^f, i = 1, 2, 3$ solved using Equation (38) (b) $\hat{f} = \sum_{i=1}^{3} \hat{a}_i^f \, \psi_i$

Fig. 6. Computation of the adjoint solution $\hat{f}$ for the Lorenz attractor. The x-axes are t; the blue, green and red lines on the left are $\hat{a}_1^f$, $\hat{a}_2^f$ and $\hat{a}_3^f$, respectively; the blue, green and red lines on the right are the first, second and third component of $\hat{f}$, computed via Equation (27).

the adjoint vector $\hat{f}$. The computed $\hat{f}$ along the trajectory is plotted both in Figure 6b as a function of t, and also in Figure 7 as arrows on the trajectory in the state space.

The last step of the adjoint sensitivity analysis algorithm is computing the sensitivity derivatives of $\langle J \rangle$ to the perturbations $\delta f_s = \frac{df}{ds}$, $\delta f_r = \frac{df}{dr}$ and $\delta f_b = \frac{df}{db}$ using Equation (26). Here our estimates over the time interval $[0, T_A]$

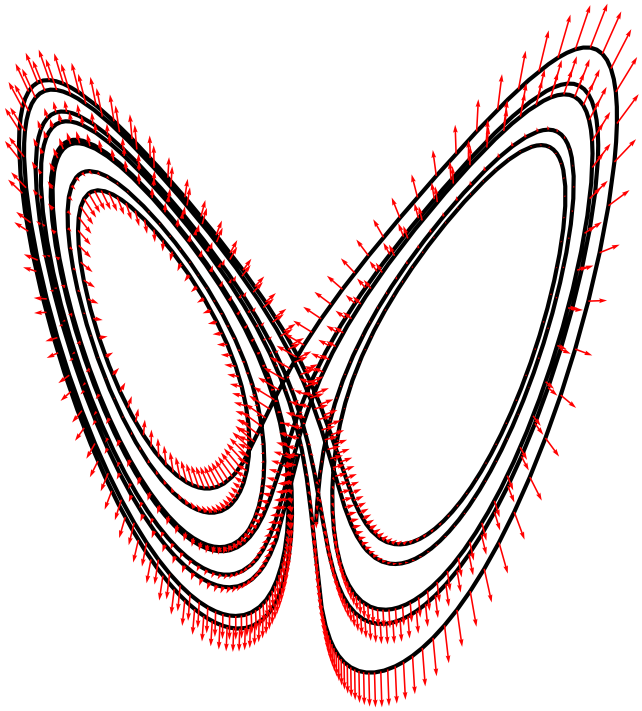


Fig. 7. The adjoint sensitivity derivative $\hat{f}$ as in Equation (26), represented by arrows on the trajectory.

are computed as

$$\frac{d\langle x_3\rangle}{ds} \approx 0.21 \ , \quad \frac{d\langle x_3\rangle}{dr} \approx 0.97 \ , \quad \frac{d\langle x_3\rangle}{db} \approx -1.74 \tag{45}$$

Note that $\frac{d\langle x_3\rangle}{dr}$ estimated using adjoint method differs from the same value estimated using forward method (44). This discrepancy can be caused by the different numerical treatments to the time dilation term in the two methods. The forward method numerically estimates the time dilation constant η through Equation (22); while the adjoint method sets the mean of $\hat{a}_i^f$ to zero (36), so that the computation is independent to the value of η. This difference could cause apparent discrepancy in the estimated sensitivity derivatives.

The next section compares these sensitivity estimates, together with the sensitivity estimates computed in Section 7.2, to a finite difference study.

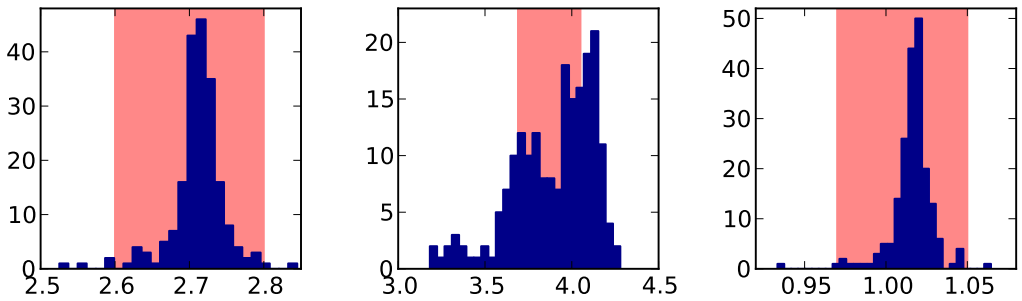
7.4 Comparison with the finite difference method

To reduce the noise in the computed statistical quantities in the finite difference study, a very long time integration length of $T = 100,000$ is used for each simulation. Despite this long time averaging, the quantities computed

contain statistical noise of the order 0.01. The noise limits the step size of the finite difference sensitivity study. Fortunately all the output statistical quantities seem fairly linear with respect to the input parameters, and a moderately large step size of the order 0.1 can be used. To further reduce the effect of statistical noise, we perform linear regressions through 10 simulations of the Lorenz attractor, with r equally spaced between 27.9 and 28.1. The total time integration length (excluding spin up time) is $1,000,000$. The resulting computation cost is in sharp contrast to our method, which involves a trajectory of only length 20.

Similar analysis is performed for the parameters s and b, where 10 values of s equally spaced between 9.8 and 10.2 are used, and 10 values of b equally spaced between $8/3-0.02$ and $8/3+0.02$ are used. The slopes estimated from the linear regressions, together with 3σ confidence intervals (where σ is the standard error of the linear regression) is listed below:

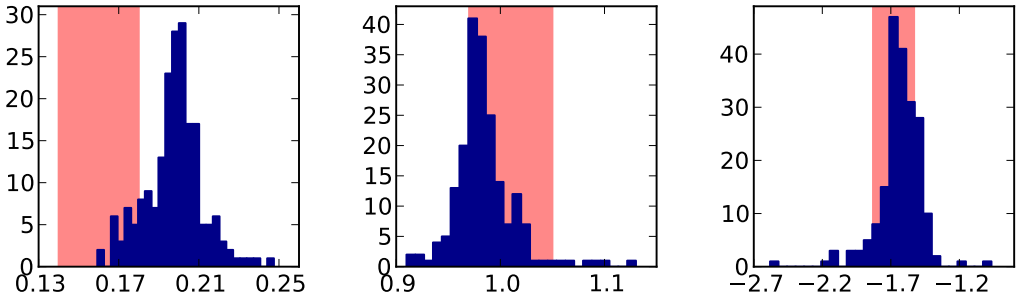
$$\begin{aligned} &\frac{d\langle x_1^2\rangle}{dr} = 2.70 \pm 0.10 \ , \quad \frac{d\langle x_2^2\rangle}{dr} = 3.87 \pm 0.18 \ , \quad \frac{d\langle x_3\rangle}{dr} = 1.01 \pm 0.04 \\ &\frac{d\langle x_3\rangle}{ds} = 0.16 \pm 0.02 \ , \quad \frac{d\langle x_3\rangle}{db} = -1.68 \pm 0.15 \ . \end{aligned} \tag{46}$$



(a) $\frac{\partial\langle x_1^2\rangle}{\partial r}$ (b) $\frac{\partial\langle x_2^2\rangle}{\partial r}$ (c) $\frac{\partial\langle x_3\rangle}{\partial r}$

Fig. 8. Histogram of sensitivities computed using Algorithm 1 (forward sensitivity analysis) starting from 200 random initial conditions. $T_A = 10, T_B = 5$. The red region identifies the 3σ confidence interval estimated using finite difference regression.

To further assess the accuracy of our algorithm, which involves finite time approximations to Equations (9) and (26), we repeated both Algorithm 1 and Algorithm 2 for 200 times, starting from random initial conditions at $T = -10$. We keep the statistical averaging time $T_A = 10$ and the spin up buffer time $T_B = 5$. The resulting histogram of sensitivities computed with Algorithm 1 is shown in Figure 8; the histogram of sensitivities computed with Algorithm 2 is shown in Figure 9. The finite difference estimates are also indicated in these plots.



(a) $\frac{\partial \langle x_3 \rangle}{\partial s}$ (b) $\frac{\partial \langle x_3 \rangle}{\partial r}$ (c) $\frac{\partial \langle x_3 \rangle}{\partial b}$

Fig. 9. Histogram of sensitivities computed using Algorithm 2 (adjoint sensitivity analysis) starting from 200 random initial conditions. $T_A = 10, T_B = 5$. The red region identifies the 3σ confidence interval estimated using finite difference regression.

We observe that our algorithms compute accurate sensitivities most of the time. However, some of the computed sensitivities seems to have heavy tails in their distribution. This may be due to behavior of the Lorenz attractor near the unstable fixed point $(0, 0, 0)$. Similar heavy tailed distribution has been observed in other studies of the Lorenz attractor [1]. They found that certain quantities computed on Lorenz attractor can have unbounded second moment. This could be the case in our sensitivity estimates. Despite this minor drawback, the sensitivities computed using our algorithm have good quality. Our algorithms are much more efficient than existing sensitivity computation methods using ensemble averages.

8 Conclusion

This paper derived a forward algorithm and an adjoint algorithm for computing sensitivity derivatives in chaotic dynamical systems. Both algorithms efficiently compute the derivative of statistical quantities $\langle J \rangle$ to infinitesimal perturbations $\epsilon\, \delta f$ to the dynamics.

The forward algorithm starts from a given perturbation δf, and computes a perturbed "shadow" coordinate system δx, e.g. as shown in Figure 1. The sensitivity derivatives of multiple statistical quantities to the given δf can be computed from δx. The adjoint algorithm starts from a statistical quantity $\langle J \rangle$, and computes an adjoint vector $\hat{f}$, e.g. as shown in Figure 7. The sensitivity derivative of the given $\langle J \rangle$ to multiple input perturbations can be computed from $\hat{f}$.

We demonstrated both the forward and adjoint algorithms on the Lorenz

attractor at standard parameter values. The forward sensitivity analysis algorithm is used to simultaneously compute $\frac{\partial\langle x_1^2\rangle}{\partial r}$, $\frac{\partial\langle x_2^2\rangle}{\partial r}$, and $\frac{\partial\langle x_3\rangle}{\partial r}$; the adjoint sensitivity analysis algorithm is used to simultaneously compute $\frac{\partial\langle x_3\rangle}{\partial s}$, $\frac{\partial\langle x_3\rangle}{\partial r}$, and $\frac{\partial\langle x_3\rangle}{\partial b}$. We show that using a single trajectory of length about 20, both algorithms can efficiently compute accurate estimates of all the sensitivity derivatives.

References

[1] G. Eyink, T. Haine, and D. Lea. Ruelle's linear response formula, ensemble adjoint schemes and Lévy flights. *Nonlinearity*, 17:1867–1889, 2004.

[2] F. Ginelli, P. Poggi, A. Turchi, H. Chaté, R. Livi, and A. Politi. Characterizing dynamics with covariant Lyapunov vectors. *Physical Review Letters*, 99:130601, Sep 2007.

[3] A. Jameson. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3:233–260, 1988.

[4] J. Krakos, Q. Wang, S. Hall, and D. Darmofal. Sensitivity analysis of limit cycle oscillations. *Journal of Computational Physics*, (0):–, 2012.

[5] D. Lea, M. Allen, and T. Haine. Sensitivity analysis of the climate of a chaotic system. *Tellus*, 52A:523–532, 2000.

[6] D. Ruelle. Differentiation of SRB states. *Communications in Mathematical Physics*, 187:227–241, 1997.

[7] D. Ruelle. A review of linear response theory for general differentiable dynamical systems. *Nonlinearity*, 22(4):855, 2009.

[8] J.-N. Thepaut and P. Courtier. Four-dimensional variational data assimilation using the adjoint of a multilevel primitive-equation model. *Quarterly Journal of the Royal Meteorological Society*, 117(502):1225–1254, 1991.

[9] D. Venditti and D. Darmofal. Grid adaptation for functional outputs: Application to two-dimensional inviscid flow. *Journal of Computational Physics*, 176:4069, 2002.

[10] Q. Wang, P. Moin, and G. Iaccarino. Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation. *SIAM Journal on Scientific Computing*, 31(4):2549–2567, 2009.

[11] C. Wolfe and R. Samelson. An efficient method for recovering Lyapunov vectors from singular vectors. *Tellus A*, 59(3):355–366, 2007.