Journal of Computational Physics 395 (2019) 690–709

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Adjoint sensitivity analysis on chaotic dynamical systems by Non-Intrusive Least Squares Adjoint Shadowing (NILSAS)

Angxiu Ni [a,*], Chaitanya Talnikar [b]

[a] Department of Mathematics, University of California, Berkeley, CA 94720, USA
[b] Nvidia Corporation, Santa Clara, CA 95051, USA

通过非侵入式平方伴随阴影 (NILSAS)
对混沌动力系统进行伴随灵敏度分析

ARTICLE INFO

Article history:
Received 3 August 2018
Received in revised form 8 April 2019
Accepted 13 June 2019
Available online 14 June 2019

Keywords:
Sensitivity analysis
Chaos
Adjoint methods
Shadowing methods
Non-intrusive formulation

敏感性分析
混沌
伴随方法
影子方法
非侵入式配方

ABSTRACT

We develop the NILSAS algorithm, which performs adjoint sensitivity analysis of chaotic systems via computing the adjoint shadowing direction. NILSAS constrains its minimization to the adjoint unstable subspace, and can be implemented with little modification to existing adjoint solvers. The computational cost of NILSAS is independent of the number of parameters. We demonstrate NILSAS on the Lorenz 63 system and a turbulent three-dimensional flow over a cylinder.

NILSAS 算法.
计算伴随阴影方向 → chaotic system 的灵敏度分析
构建最小二乘问题
演示 NILSAS 的应用
计算梯度

1. Introduction

Sensitivity analysis is a powerful tool in helping scientists and engineers design products [1], control processes and systems [3,4], solve inverse problems [5], estimate simulation errors [6–9], assimilate measurement data [10,11] and quantify uncertainties [12]. Adjoint sensitivity analysis computes the gradient of one objective with respect to many parameters in one run, so it can be very useful for cases with many parameters, such as geometry design [2] and training neural networks [13,14].

Conventional sensitivity analysis works well for systems with a stable fixed point or a periodic orbit. However, when the system is chaotic and the design objective is a long-time-averaged quantity, conventional sensitivity methods, be it tangent or adjoint, fails to provide useful sensitivity information. Chaotic dynamical systems are typically modeled as hyperbolic systems, for which many sensitivity analysis methods have been developed, such as by Lea et al. in [15,16], by Abramov and Majda in [17,18], and by Lucarini et al. in [19,20].

Sensitivity of hyperbolic dynamical systems can be computed via the shadowing direction. Bowen [21] proved that for a trajectory of a uniform hyperbolic systems, if we introduce a small perturbation in the governing equation, there exists a shadowing trajectory, which satisfies the perturbed equations, but still lies close to the base trajectory. Later Pilyugin [22] gave a formula for the first order difference between the shadowing trajectory and the base trajectory. In this paper we call such first order difference the shadowing direction. The Least Squares Shadowing (LSS) approach, developed by Wang et al., [23–25], computes such shadowing direction through an L^2 minimization, and use it for sensitivity analysis.

不动点
周期轨道
双曲动态系统
轨道
扰动方程

1. Lea等人贡献于混沌系统建模和敏感性分析。
2. Abramov和Majda专注于混沌动力系统的敏感性分析方法。
3. Lucarini等人在混沌系统建模和敏感性分析方面有贡献。
4. Bowen证明了均匀双曲系统轨迹性质，引入小扰动后仍有阴影轨迹靠近基本轨迹。
5. Pilyugin提出了阴影轨迹和基本轨迹一阶差的公式。
6. Wang等人通过最小二乘阴影(LSS)方法在混沌系统敏感性分析方面有贡献。

* Corresponding author.
E-mail addresses: niangxiu@gmail.com, niangxiu@math.berkeley.edu (A. Ni), chaitukca@gmail.com (C. Talnikar).
URL: https://math.berkeley.edu/~niangxiu/ (A. Ni).

https://doi.org/10.1016/j.jcp.2019.06.035
0021-9991/

Sensitivity Analysis: 敏感度分析
传统: 稳定/周期 ✓
混沌/瞬时间.平均值 ×
→ 最小二乘阴影 (LSS): 通过数学优化计算阴影方向.

介绍 NILSAS 的方法（同 abstract）
并将最小化问题限制在不稳定伴随子空间中（成本低，收敛性好）

The Non-Intrusive Least Squares Shadowing method (NILSS) developed by Ni et al. [26,27] finds a 'non-intrusive' formulation of the LSS problem which constrains the minimization problem in LSS to the unstable subspace. For many real-life problems, the dimension of unstable subspace is much lower than the dimension of the dynamical system, and NILSS can be thousands times faster than LSS. A major variant of NILSS is the Finite Difference NILSS (FD-NILSS) algorithm [28], whose implementation requires only primal solvers, but not tangent solvers. FD-NILSS has been applied to several complicated flow problems [29,28] which were too expensive for previous sensitivity analysis methods.

The marginal cost for a new parameter in NILSS is only computing one extra inhomogeneous tangent solution. Yet for cases where there are many parameters and only a few design objectives, an adjoint version of NILSS is desired. A continuous adjoint version appeared in the first publication of NILSS [26]; however, this version of lacks the constraint on the neutral subspace, which will be explained in our current paper. Blonigan [30] developed a discrete adjoint version of NILSS, which was later implemented by Chandramoorthy et al. [31] using automatic differentiation. In comparison to this discrete adjoint NILSS, NILSAS does not require tangent solvers, and requires less modification to existing adjoint solvers, and the simplicity of the formula of NILSAS should also give it more robustness and perhaps better convergence.

Recently, we defined the adjoint shadowing direction for both hyperbolic flows and diffeomorphisms, which is a bounded inhomogeneous adjoint solution with several other properties [32]. We showed that the adjoint shadowing direction exists uniquely on a given trajectory, and can be used for adjoint sensitivity analysis. Adjoint shadowing direction is defined using only adjoint flows, giving us a chance to get rid of tangent solvers in our algorithm, and arrive at a neat formula.

This paper presents the Non-Intrusive Least Squares Adjoint Shadowing (NILSAS) algorithm, where we construct a least squares problem to approximate the adjoint shadowing direction and then compute adjoint sensitivity. The 'non-intrusive' formulation of NILSAS allows it is built using existing adjoint solvers, and more importantly, it allows the minimization be constrained to the unstable adjoint subspace. The main body of this paper will be about continuous dynamical systems, and we briefly discuss NILSAS for discrete systems in Appendix B.

We organize the rest of this paper as follows. First, we prepare our study by defining our problem and reviewing adjoint flows and adjoint shadowing directions; we also provide some intuitions to help understanding adjoint shadowing directions. Then we derive the NILSAS algorithm. Then we present a detailed procedure list for our algorithm and give several remarks on the algorithm. Finally, we demonstrate NILSAS on the Lorenz 63 system and a weakly turbulent three-dimensional (3D) flow over a cylinder.

2. Preparations

控制方程（描述系统/行为） 时间演化混沌过程

The governing equation of a hyperbolic flow, which models a time-evolving chaotic process, is:

$$\frac{du}{dt} = f(u,s), \quad u(t=0) = u_0. \tag{1}$$

初状态 状态 参数

混沌是一种对初始条件极其敏感、表现出复杂无序、非周期性的行为的系统状态。在时间演化的过程中，混沌系统的状态会在相空间中不断变化，而且即使微小的初始差异也可能导致系统轨迹迅速发散，最终表现出复杂而看似随机的运动。

This differential equation is called the primal system, and a solution $u(t)$ is the primal solution. Here $f(u,s): \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^m$ is a smooth function, $u \in \mathbb{R}^m$ is the state of the dynamical system, u_0 is the initial condition, and $s \in \mathbb{R}$ is the parameter. For now we assume there is only one parameter; and in section 4.2.1 we will explain how we can compute sensitivities with respect to several parameters with almost no additional cost.

长时间平均量

In this paper, we assume there is only one objective, which is a long-time-averaged quantity. To define it, we first let $J(u,s): \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}$ be a continuous function that represents the instantaneous objective. The objective is obtained by averaging over a semi-infinite trajectory:

瞬时值

$$J_{avg} := \lim_{T\to\infty} \frac{1}{T} \int_0^T J(u,s)dt. \tag{2}$$

半无穷轨迹

原系统（Primal System）：初始模型（问题的主要表述）
伴随系统（Adjoint System）：补充（用于敏感性分析/优化）

We assume the system has a global attractor [25], hence J_{avg} only depends on s. Our goal of this paper is to develop an algorithm computing the sensitivity, dJ_{avg}/ds, whose marginal cost for a new parameter is negligible.

2.1. Adjoint flow

计算 sensitivity，即 $\frac{dJ_{avg}}{ds}$

Definition 1. A homogeneous adjoint solution $w(t): \mathbb{R} \to \mathbb{R}^m$ is a function which solves the homogeneous adjoint equation:

齐次伴随解

$$\frac{dw}{dt} + f_u^T w = 0, \tag{3}$$

where $\cdot^T$ is the matrix transpose. An inhomogeneous adjoint solution is a function $v(t): \mathbb{R} \to \mathbb{R}^m$ which solves:

非齐次伴随解

$$\frac{dv}{dt} + f_u^T v = g(t), \tag{4}$$

t：时间

where $g(t): \mathbb{R} \to \mathbb{R}^m$ is a vector-valued function of time.

通常用时间逆序求解，否则 unstable subspace 维度过大

采纳

In numerical implementations, we typically solve adjoint equations backward in time. This is because, as shown in [32], when solving backward in time, the dimension of the unstable adjoint subspace is the same as the unstable tangent subspace, which is typically much lower than m. On the other hand, if we solve the adjoint equation forward in time, the unstable subspace has much higher dimension, causing strong numerical instability.

Definition 2. In this paper, an adjoint covariant Lyapunov vector (CLV) with adjoint Lyapunov exponent (LE) λ is a homogeneous adjoint solution $\overline{\zeta}(t)$ such that there is a constant C, for any $t_1, t_2 \in \mathbb{R}$,

$$\|\overline{\zeta}(t_1)\| \leq Ce^{\lambda(t_2-t_1)}\|\overline{\zeta}(t_2)\| . \tag{5}$$

Note that the time direction in the above definition is reversed: if the adjoint CLV grows exponentially backward in time, its exponent is positive. Adjoint CLVs with positive exponents are called unstable, those with negative exponents are stable, and those with zero exponent is been neutral. In this paper, we sort adjoint CLVs by descending order of their exponents. The earliest mention of adjoint CLVs was by Kuptsov and Parlitz in [33]. For the purpose of defining adjoint shadowing directions and deriving the NILSAS method, we recently proved the existence of adjoint CLVs under the same assumptions of adjoint shadowing theorem, and found some relation between CLVs and adjoint CLVs.

Adjoint CLVs are homogeneous adjoint solutions whose norm grows exponentially, and the adjoint LE is measured backward in time. This is similar but also different from tangent CLVs, which are tangent solutions measured forward in time. The CLV structure for the adjoint flow is the same as the tangent flow. That is, the adjoint LE spectrum is the same as the tangent LE spectrum. Moreover, the subspace of CLVs with an exponent λ is perpendicular to the subspace of all adjoint CLVs with exponents not λ, and vice versa. If we can write the full set of CLVs as a matrix valued function of time, $W(t)$, then $W^{-T}(t)$, where $\cdot^{-T}$ is the inverse of transpose, is a matrix whose columns are adjoint CLVs: readers can verify that $W^{-T}(t)$ satisfies the properties listed above. Note that we do not know if CLVs and adjoint CLVs with the same exponent are perpendicular or parallel.

We assume our system is uniform hyperbolic and it has a bounded global attractor. Definition of hyperbolicity can be found in most textbook on dynamical system such as [34], and readers may also refer to [32] for a definition using the same notation as this paper. Uniform hyperbolicity requires that the tangent space can be split into stable subspace, unstable subspace, and a neutral subspace of dimension one. Together with the boundedness of the attractor, we can show the angles between two subspaces of different sets of tangent CLVs are always larger than some positive angle. Since the adjoint equations have the same structure as the tangent ones, there is only one neutral adjoint CLV, and adjoint CLVs are always bounded away from each other [32].

2.2. Adjoint shadowing directions

In [32], the author defined adjoint shadowing directions, proved their unique existence on a given trajectory, and showed how to use them for adjoint sensitivity analysis. We briefly restate the main results in this subsection.

Definition 3. On a trajectory $u(t)$ on the attractor, for $t \geq 0$, the adjoint shadowing direction $\overline{v}^{\infty} : \mathbb{R}_+ \to \mathbb{R}^m$ is defined as a function with the following properties:

1. $\overline{v}^{\infty}$ solves the inhomogeneous adjoint equation:
$$\frac{d\overline{v}^{\infty}}{dt} + f_u^T \overline{v}^{\infty} = -J_u , \tag{6}$$
where subscripts are partial derivatives, that is, $f_u = \partial f/\partial u$, $J_u = \partial J/\partial u$.
2. $\overline{v}^{\infty}(t=0)$ has zero component in the unstable adjoint subspace.
3. $\|\overline{v}^{\infty}(t)\|$ is bounded by a constant for all $t \in \mathbb{R}_+$.
4. The averaged inner-product of $\overline{v}^{\infty}$ and f is zero:
$$\left\langle \overline{v}^{\infty}, f \right\rangle_{avg} := \lim_{T\to\infty} \frac{1}{T} \int_0^T \left\langle \overline{v}^{\infty}(t), f(t) \right\rangle = 0 , \tag{7}$$
where $\langle \cdot, \cdot \rangle$ is the inner-product on the Euclidean space.

We remind readers to distinguish the three different kinds of adjoint solutions we mentioned: homogeneous adjoint solutions, inhomogeneous adjoint solutions and adjoint shadowing directions. Homogeneous adjoint solutions are different from inhomogeneous ones, since homogeneous adjoint equations must have zero right-hand-sides. The adjoint shadowing direction is an inhomogeneous adjoint solution, but not any inhomogeneous adjoint solution: it must in extra have three more properties listed in the definition. In fact, one way to view NILSAS is that we search the space of all inhomogeneous adjoint solutions to find one such that it mimics the other three properties. More specifically, we minimize the L^2 norm, and constrain the inner product with f: this derivation will be revealed in later sections.

Theorem 1. *For a uniform hyperbolic system with a global compact attractor, on a trajectory on the attractor, there exists a unique adjoint shadowing direction. Further, we have the adjoint sensitivity formula:*

$$\frac{dJ_{avg}}{ds} = \lim_{T\to\infty} \frac{1}{T} \int_0^T \langle \overline{v}^{\infty}, f_s \rangle + J_s \, dt \,. \tag{8}$$

We explain the assumption of the adjoint shadowing theorem. First, if a dynamical system has a compact global attractor, it means there is a bounded set of states, or the attractor, such that no matter what initial condition the system starts from, the trajectory will eventually enter the attractor and never leave. Second, uniform hyperbolicity here mainly means that there is only one neutral CLV. Third, by the compactness, the angles between all CLVs are larger than a positive angle, regardless of where we are on the attractor.

Why do we make above assumptions in theories for shadowing methods? The main reason for assuming only one neutral CLV in shadowing methods is to prevent linear growth in inhomogeneous tangent/adjoint solutions. The main reason for global attractability is to ensure that shadowing trajectories are representative of the averaged behavior of the system. The main reason for compactness is because we want a bound for the projection operators projecting onto a particular subspace. Still, we remind readers that, in practice, shadowing methods may be effective beyond above assumptions, as to be discussed in section 4.2.1.

Rather than giving an explicit expression of adjoint shadowing directions, which can be found as well in [32], the definition is stated as a criterion, where we check several properties to determine if a function is indeed the adjoint shadowing direction. In fact, we forged this definition for designing the NILSAS algorithm, which will be revealed in the next section.

2.3. Interpreting adjoint shadowing directions

We give several different perspectives to help readers build intuitions on adjoint shadowing directions. To start with, we revisit some formal descriptions of shadowing operators. The shadowing operator, denoted as $\mathcal{S}$, can be viewed roughly as mapping a vector-valued function $f_s(t)$ to another vector-valued function $v^{\infty}(t)$, Here $f_s(t)$ is the perturbation on f due to parameter perturbations; v^{∞} is the (tangent) shadowing direction, which is first order approximation of the difference between the shadowing and the base trajectory. Note that $\mathcal{S}$ is a linear operator, and both $f_s(t)$ and $v^{\infty}(t)$ are linear approximations. If we neglect the subtleties due to the neutral CLV, we have roughly

$$\frac{dJ_{avg}}{ds} \approx \left\langle v^{\infty}, J_u \right\rangle_{avg} = \langle \mathcal{S}(f_s), J_u \rangle_{avg} \,. \tag{9}$$

Where $\langle \cdot, \cdot \rangle_{avg}$ is an inner product.

First we provide a utility point of view for adjoint shadowing directions, which is also an algebraic point of view. Riesz's representation theorem tells us that there is an adjoint operator $\overline{\mathcal{S}}$ such that

$$\langle \mathcal{S}(f_s), J_u \rangle_{avg} = \left\langle f_s, \overline{\mathcal{S}}(J_u) \right\rangle_{avg} \,. \tag{10}$$

The adjoint shadowing direction, $\overline{v}^{\infty}$, can be viewed as $\overline{\mathcal{S}}(J_u)$. Suppose now that we have a computer program which approximately functions as $\overline{\mathcal{S}}$. If we have two parameters s_1, s_2, then they can perturb the governing equation by f_{s_1} and f_{s_2}, whereas J_u keeps the same. This means that we only need to run our adjoint program once, and use the result to inner-product with both f_{s_1} and f_{s_2}. For cases where there are many parameters s and a few objectives J, we only need to run our program a few times.

Then an implementation point of view. The adjoint operator of a matrix is simply its transpose. For many other cases, we can see that matrix transposition often appears as a crucial step in the formulation of adjoint operators. So once we derive an new adjoint formula of something, we may ask if it can be presented as neatly as transposing a matrix. Adjoint shadowing directions satisfy inhomogeneous adjoint equations, which is a linear ODE whose matrix is the transpose of the Jacobian matrix. Hence there is chance that algorithms, such as NILSAS, do not differ too much from existing adjoint solvers. In fact, giving a neat recipe for adjoint shadowing direction is the main contribution of both the adjoint shadowing theorem and the NILSAS algorithm, since the existence is already given by Riesz's representation theorem.

Finally, we provide a physical point of view. Assume the initial condition of our trajectory is fixed, and we perturb the state at t by δu_t, then the trajectory at a later time, $\tau \geq t$, is changed by $D_t^{\tau} \delta u_t$. Now the change in J_{avg} satisfies:

$$T(J_{avg} + \delta J_{avg}) = \int_0^T J(u(\tau) + \delta u(\tau), s) d\tau = \int_0^T J(u, s) d\tau + \int_t^T J_u^T D_t^{\tau} \delta u_t d\tau \,.$$

Here J_u is transposed to comply with our notation that J_u is a column vector. Canceling the first term on both sides and diving both sides by δu_t, we get

$$T\frac{\delta J_{avg}}{\delta u_t} = \int_t^T J_u^T D_t^\tau d\tau \,. \tag{11}$$

This is exactly $\overline{v}^*(t)$, the conventional adjoint solution at t.

Instead of assuming initial condition being fixed, we let it change according to that prescribed by the shadowing direction. Now $T\delta J_{avg}/\delta u_t$ is the adjoint shadowing solution $\overline{v}^\infty(t)$. This means that the adjoint shadowing direction can be viewed as how a perturbation at t in the trajectory affects the objective, with the caveat that now both the past and future trajectory are perturbed to match the perturbation at t.

3. Deriving NILSAS

3.1. The non-intrusive formulation

On a finite trajectory of time span $[0, T]$, the NILSAS algorithm computes a $\overline{v}$ which approximates $\overline{v}^\infty$. Since the definition of the adjoint shadowing direction is similar to the tangent shadowing direction, it is not surprising that we can reuse the 'non-intrusive' formulation in NILSS, that is, we can find adjoint shadowing direction by a minimization in the unstable adjoint subspace.

In NILSAS, we strictly enforce the first property of adjoint shadowing directions by constraining our solutions to inhomogeneous adjoint solutions. The second property is changed to a symmetric statement that stable component in $\overline{v}(T)$ should be $O(1)$, which can be easily satisfied. The third property is approximated by minimizing the L^2 norm of the inhomogeneous adjoint solution. The fourth property is strictly enforced by adding a constraint to our minimization problem. In this subsection, we explain why the $\overline{v}$ given by this reverse-engineering is a good approximation of $\overline{v}^\infty$.

Our algorithm strictly enforces the first property of $\overline{v}^\infty$. To do this, we represent the solution set of equation (6) as a particular solution plus the space of homogeneous solutions. We select the particular solution as the conventional inhomogeneous adjoint solution $\overline{v}^*$, which is defined as the solution of:

$$\frac{d\overline{v}^*}{dt} + f_u^T\overline{v}^* = -J_u \,,\ \overline{v}^*(T) = 0 \,. \tag{12}$$

Then we select the collection of all adjoint CLVs, $\overline{Z} = [\overline{\zeta}_1, \cdots, \overline{\zeta}_m]$, all of which have terminal condition $\|\overline{\zeta}_j(T)\| = 1$, as the basis of the space of homogeneous solutions. Hence we can enforce the first property by considering candidates only in the following form for some $\{a_j\}_{j=1}^m$:

$$\overline{v} = \overline{v}^* + \sum_{j=1}^m a_j\overline{\zeta}_j = \overline{v}^* + \overline{Z}a \,, \tag{13}$$

where the coefficients $a = [a_1, ..., a_m]^T$ is a column vector. Another interpretation of our way of enforcing the first property is that, we want to start $\overline{v}$ from $\overline{v}^*$, and modify by adding adjoint CLVs to approximate $\overline{v}^\infty$; in other words, the coefficients a should be such that $Za \approx \overline{v}^\infty - \overline{v}^*$. We then use other properties to determine the coefficients for stable, unstable and neutral CLVs.

When defining adjoint shadowing directions in [32], the author was considering functions defined starting from time zero, whereas in our case here, adjoint solutions are solved from T backward in time. Hence, in order to keep the sensitivity formula, we change the second property to a symmetric statement, that is, we want the stable component in $\overline{v}(T)$ be in the order $O(1)$, meaning be bounded by a constant independent of T. Equivalently, we require coefficients for stable CLVs be $O(1)$. Another way to interpret is that, if this $O(1)$ condition is true, then since $\overline{v}^\infty(T)$ is $O(1)$, the stable component in $\overline{v}(T) - \overline{v}^\infty(T)$ is $O(1)$; now since stable CLVs decay exponentially fast, their contribution in $\overline{v} - \overline{v}^\infty$ can be neglected. This $O(1)$ condition is a loose requirement and, as we will see, it can be easily satisfied.

To mimic the boundedness in the third property of adjoint shadowing directions, we minimize $\|\overline{v}\|_{L^2}$, which determines the coefficients for the unstable CLVs. Indeed, this minimization removes significant unstable CLVs from $\overline{v} - \overline{v}^\infty$, since otherwise this difference would grow exponentially, and since $\overline{v}^\infty$ is bounded, $\overline{v}$ would have large L^2 norm.

We strictly enforce the fourth property of $\overline{v}$. This determines the coefficient for the neutral adjoint CLV, since as shown in [32], f is always orthogonal to non-neutral adjoint CLVs. Note also that the norm of the neutral adjoint CLV is bounded, unlike neutral tangent CLV, which can have linear growth. Hence we can allow its coefficient be $O(1)$ without jeopardizing the boundedness property we used earlier. In fact, the adjoint NILSS in [26] lacks exactly this constraint on the neutral adjoint CLV.

To summarize, we determine coefficients for unstable adjoint CLVs via a minimization, the coefficient for the neutral adjoint CLV via equation (7), and we do not care coefficients for stable adjoint CLVs too much. Hence there is no need to provide stable CLVs to our algorithm; it is even unnecessary to provide accurate non-stable CLVs, they can contain some stable components at T. Further, we care not individual CLVs but only their span. Hence we can replace $\overline{Z}$ by $\overline{W} = [\overline{w}_1, \cdots, \overline{w}_M]$, with $M \geq m_{us} + 1$, m_{us} being the number of unstable CLVs, and $\{\overline{w}_j\}_{j=1}^M$ are homogeneous adjoint solutions

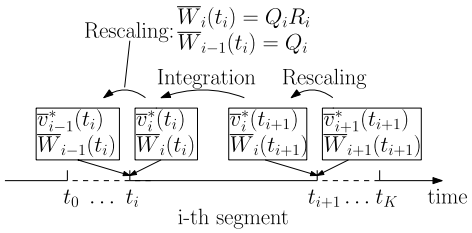


Fig. 1. Notations for multiple segments. $\overline{W}_i(t), \overline{v}^*_i(t)$ are defined on the i-th segment, which spans $t \in [t_i, t_{i+1}]$; Q_i, R_i are defined at t_i. 'Integration' refers to integrating adjoint equations for $\overline{W}_i(t), \overline{v}^*_i(t)$: after this procedure we move from end to the start within one segment. 'Rescaling' refers to renormalize adjoint solutions at the interface between segments: after this procedure we move to another time segment.

whose non-stable components at T span the entire non-stable subspace. Such a set of solutions can be obtained by solving homogeneous adjoint equations from almost all terminal conditions of M randomized unit vectors.

With above discussions, we see that our algorithm should solve the NILSAS problem on one segment:

$$\begin{aligned} &\min_{a\in\mathbb{R}^M} \frac{1}{2}\int_0^T \left\langle \overline{v}^* + \overline{W}a, \overline{v}^* + \overline{W}a \right\rangle , \\ &\quad \text{s.t.} \int_0^T \left\langle \overline{v}^* + \overline{W}a, f \right\rangle = 0 . \end{aligned} \tag{14}$$

This is simply a least squares problem with arguments $a \in \mathbb{R}^M$. Note that all adjoint solutions can be computed after making little modifications to existing adjoint solvers, and our minimization is constrained to essentially only the unstable adjoint subspace: these are the benefits of the non-intrusive formulation. Letting $\overline{v} = \overline{v} + \overline{W}a$, we can compute sensitivity via equation (8) on a finite trajectory:

$$\frac{dJ_{avg}}{ds} \approx \frac{1}{T}\int_0^T \left\langle \overline{v}, f_s \right\rangle + J_s \, dt . \tag{15}$$

3.2. Dividing trajectory into segments

An issue in numerical stability is that, as the trajectory gets longer, all adjoint solutions become dominated by the fastest growing adjoint CLV; as a result, the minimization problem in equation (14) becomes ill-conditioned. This issue also happened in NILSS [27,26] and in the algorithm for computing CLVs [35], and here we use a similar technique to resolve it, that is, dividing the whole trajectory into multiple segments, and rescaling at interfaces.

Roughly speaking, at the end of each segment, we orthogonalize and rescale adjoint solutions $\overline{W}$ and $\overline{v}^*$ so that they are no longer dominated by the first CLV. Note here since adjoint solutions are integrated backward in time, the initial condition is at the end of a segment. Despite that now $\overline{W}$ and $\overline{v}^*$ are discontinuous across segments, we can still construct $\overline{v}$ as their linear combinations, and keep $\overline{v}$ continuous across all segments. This continuous $\overline{v}$ computed from multiple segments should be identical to that solved on one large segment containing the entire trajectory.

We first define some notations, as shown in Fig. 1. Let T be the time length of the entire trajectory, and K the total number of segments. We denote the time span of the i-th segment by $[t_i, t_{i+1}]$, where $t_0 = 0, t_K = T$. For quantities defined on a entire segment such as $\overline{W}_i, \overline{v}^*_i$, C_i, d_i and a_i, their subscripts are the same as the segment they are defined on. For quantities defined only at the interfaces between segments such as Q_i, R_i, b_i, p_i and λ_i, their subscripts are the same as the time point they are defined at. Some of the notations are used immediately below, the others will be used in section 4.1 and Appendix A.

At time t_i, we perform QR factorization to $\overline{W}_i(t) = [\overline{w}_{i1}(t), \cdots \overline{w}_{iM}(t)]$, which is a $m \times M$ matrix whose column vectors are homogeneous adjoints on segment i. We use the Q-matrix, the matrix with orthonormal columns, as the terminal condition for $\overline{W}_{i-1}$ on segment $i-1$. More specifically,

$$\overline{W}_i(t_i) = Q_i R_i, \quad \text{and} \quad \overline{W}_{i-1}(t_i) = Q_i . \tag{16}$$

At time t_i, we also rescale $\overline{v}^*_i(t)$, which is the particular inhomogeneous adjoint solution on segment i. To do this, we subtract from $\overline{v}^*_i$ its orthogonal projection onto homogeneous adjoint solutions. More specifically,

$$p_i := \overline{v}^*_i(t_i) - Q_i b_i , \text{ where } b_i = Q_i^T \overline{v}^*_i(t_i) , \text{ and } \overline{v}^*_{i-1}(t_i) = p_i . \tag{17}$$

This rescaling maintains the continuity of the affine space $\overline{v}_i^* + span\{\overline{w}_{ij}\}_{j=1}^M$ across different segments.

The continuity of affine space allows us to impose continuity condition for $\overline{v}_i$, which is the adjoint shadowing direction on segment i. On each segment, $\overline{v}_i = \overline{v}_i^* + \overline{W}_i a_i$ for some $a_i \in \mathbb{R}^M$. The continuity condition can now be expressed via a relation between a_i and a_{i-1}:

$$\overline{v}_i^*(t_i) + \overline{W}_i(t_i)a_i = \overline{v}_{i-1}^*(t_i) + \overline{W}_{i-1}(t_i)a_{i-1}\,. \tag{18}$$

Apply equation (16) and (17), cancel $\overline{v}_i^*(t_i)$ on each side, we get:

$$Q_i R_i a_i = -Q_i b_i + Q_i a_{i-1} \tag{19}$$

Since Q_i has orthonormal columns, $Q_i^T Q_i = I \in \mathbb{R}^{M\times M}$. Multiplying Q_i^T to the left of both sides, we have the continuity condition for $\overline{v}$:

$$a_{i-1} = R_i a_i + b_i\,. \tag{20}$$

4. The NILSAS algorithm

4.1. Procedure list of the algorithm

Now we give a procedure list of the NILSAS algorithm. To start with, we need to have an inhomogeneous adjoint solver and a homogeneous adjoint solver, both can take arbitrary terminal conditions. The inhomogeneous adjoint equation we solve in NILSAS has right-hand-side $-J_u$, which is the same as many existing adjoint solvers. Hence for inhomogeneous adjoint solvers in NILSAS, we only need to change existing solvers to be able to take arbitrary terminal conditions. For homogeneous adjoint solvers, we only need to further change the right-hand-side of existing solvers to zero.

We provide the following data to NILSAS: 1) the number of homogeneous adjoint solutions, $M \geq m_{us} + 1$, where m_{us} is the number of unstable CLVs; 2) the total number of segments, K; 3) for convenience, we assume that the length of all time segments are the same, denoted by ΔT. The total time length is determined by $T = K\Delta T$. Moreover, in the procedure list below, inner products are written in matrix notations, and by default vectors are in column forms.

1. Integrate the primal system for sufficiently long time before $t = 0$ so that $u(t = 0)$ is on the attractor.
2. Compute the trajectory $u(t), t \in [0, T]$, by integrating the primal system.
3. Generate terminal conditions for $\overline{W}_i$ and $\overline{v}_i^*$ on the last segment $i = K - 1$:
   (a) Randomly generate a $m \times M$ full rank matrix, Q'. Perform QR factorization: $Q_K R_K = Q'$.
   (b) Set $p_K = 0$.
4. Compute $\overline{W}_i$ and v_i^* on all segments. For $i = K - 1$ to $i = 0$ do:
   (a) To get $\overline{W}_i(t)$, whose columns are homogeneous adjoint solutions on segment i, solve:

$$\frac{d\overline{W}_i}{dt} + f_u^T \overline{W}_i = 0, \quad \overline{W}_i(t_{i+1}) = Q_{i+1}\,. \tag{21}$$

   To get $\overline{v}_i^*(t)$, solve the inhomogeneous adjoint equation:

$$\frac{d\overline{v}_i^*}{dt} + f_u^T \overline{v}_i^* = -J_u\,, \quad \overline{v}_i^*(t_{i+1}) = p_{i+1}\,. \tag{22}$$

   (b) Compute the following integrations.

$$\begin{aligned}
C_i &= \int_{t_i}^{t_{i+1}} \overline{W}_i^T \overline{W}_i dt\,, \quad d_i^{wv^*} = \int_{t_i}^{t_{i+1}} \overline{W}_i^T \overline{v}^* dt\,,\\
d_i^{wf} &= \int_{t_i}^{t_{i+1}} \overline{W}_i^T f dt\,, \quad d_i^{v^*f} = \int_{t_i}^{t_{i+1}} \overline{v}^{*T} f dt\,,\\
d_i^{wf_s} &= \int_{t_i}^{t_{i+1}} \overline{W}_i^T f_s dt\,, \quad d_i^{v^*f_s} = \int_{t_i}^{t_{i+1}} \overline{v}^{*T} f_s dt\,,\\
d_i^{J_s} &= \int_{t_i}^{t_{i+1}} J_s dt\,,
\end{aligned} \tag{23}$$

   where $d_i^{wv^*}, d_i^{wf}, d_i^{wf_s} \in \mathbb{R}^M$; $d_i^{v^*f}, d_i^{v^*f_s}, d_i^{J_s} \in \mathbb{R}$; $C_i \in \mathbb{R}^{M\times M}$ is the covariant matrix.

(c) Orthonormalize homogeneous adjoint solutions via QR factorization:

$$Q_i R_i = \overline{W}_i(t_i) \tag{24}$$

(d) Rescale the inhomogeneous adjoint solution using Q_i:

$$p_i = \overline{v}_i^*(t_i) - Q_i b_i\,, \quad \text{where } b_i = Q_i^T \overline{v}_i^*(t_i)\,. \tag{25}$$

5. Compute the adjoint shadowing direction $\{\overline{v}_i\}_{i=0}^{K-1}$.
(a) Solve the NILSAS problem on multiple segments:

$$\begin{aligned} &\min_{a_0,\cdots,a_{K-1}\in\mathbb{R}^M} \sum_{i=0}^{K-1} \frac{1}{2}(a_i)^T C_i a_i + (d_i^{wv^*})^T a_i, \quad \text{s.t.} \\ &a)\ a_{i-1} = R_i a_i + b_i\,, \quad i = 1,\cdots,K-1\,, \\ &b)\ \sum_{i=0}^{K-1} (d_i^{wf})^T a_i + \sum_{i=0}^{K-1} d_i^{v^* f} = 0\,. \end{aligned} \tag{26}$$

This is a least squares problem in $\{a_i\}_{i=0}^{K-1} \subset \mathbb{R}^M$. In Appendix A we suggest one way to solve this problem.
(b) On each time segment i, $\overline{v}_i$ is given by

$$\overline{v}_i(t) = \overline{v}_i^*(t) + \overline{W}_i(t) a_i. \tag{27}$$

6. Compute the derivative by:

$$\frac{dJ_{avg}}{ds} \approx \frac{1}{T}\sum_{i=0}^{K-1} \int_{t_i}^{t_{i+1}} \left(\overline{v}_i^T f_s + J_s\right) dt = \frac{1}{T}\sum_{i=0}^{K-1} \left(d_i^{v^* f_s} + a_i^T d_i^{wf_s} + d_i^{J_s}\right) \tag{28}$$

4.2. Remarks about NILSAS

4.2.1. Miscellaneous

We remark that if we are not interested in obtaining $\overline{v}(t)$ for all t, there is no need to store adjoint solutions $\overline{W}$ and $\overline{v}^*$ in computers, which is typically lots of data. To compute sensitivity, we only need to store $d_i^{wv^*}$, d_i^{wf}, $d_i^{wf_s}$, $d_i^{v^* f}$, $d_i^{v^* f_s}$, $d_i^{J_s}$, C_i given in equation (23), R_i given in equation (24), and b_i given in equation (25).

Moreover, when computing quantities in equation (23), we can estimate the integration using particular values of the integrands evaluated at several snapshots, to further reduce the storage management cost. For example, similar to [30], we can estimate C_i by the terminal value of W_i, which is Q_i, and thus $C_i = I \in \mathbb{R}^{M\times M}$. We suggest further research be done to determine which estimation is best practice.

NILSAS has the benefit of typical adjoint algorithms, that is, for a new parameter s, $\overline{v}$ does not change, so we only need to give new f_s, J_s, and recompute equation (23) and (28). Hence the extra cost for a new parameter is only performing an L^2 inner product, which is negligible in comparison with the total cost of the algorithm. More specifically, assume there are n parameters $s = [s_1,\cdots,s_n]$, we can define

$$\begin{aligned} \frac{dJ_{avg}}{ds} &= [\frac{\partial J_{avg}}{\partial s_1},\cdots,\frac{\partial J_{avg}}{\partial s_n}] \in R^{1\times n}\,; \\ J_s &= [\frac{\partial J}{\partial s_1},\cdots,\frac{\partial J}{\partial s_n}] \in R^{1\times n}\,; \\ f_s &= [\frac{\partial f}{\partial s_1},\cdots,\frac{\partial f}{\partial s_n}] \in R^{m\times n}\,. \end{aligned} \tag{29}$$

With these definitions, the NILSAS algorithm, in particular equation (23) and (28), extend to several parameters with almost no extra cost. An extreme example is where f_s is unknown a priori and we can now use $\overline{v}$ to design an optimal control, f_s.

The assumptions in Theorem 1 are made for theoretically proving the unique existence of adjoint shadowing directions and convergence of NILSAS. In practice, it is possible that NILSS/NILSAS are still valid on a chaotic system which fails these assumptions. For example, the 3D cylinder flow we investigate later in this paper has at least two neutral CLVs, corresponding to translations in time and in the span-wise directions, due to the periodic boundary condition. In fact, in [29] we also showed that the smallest angle between tangent CLVs depends on meshes and may fall below a threshold value: this further violates our assumptions. However, we did found the trend that angles between tangent CLVs gets larger

when their indices are further apart: this property is related to hyperbolicity, but has not been well investigated yet. As we shall see, both NILSS and NILSAS compute correct sensitivities on this 3D flow. The generality of shadowing methods is as suggested by the chaotic hypothesis [36,37], that is, theoretical tools may still valid for non-uniform hyperbolic chaotic systems, even though those tools can only be rigorously proved with a stricter assumption. We do not expect NILSS and NILSAS be valid for all chaotic systems; however, they are valid somewhere beyond our current assumptions. We call for more research to identify the limit of shadowing methods, especially in real-life problems.

In Appendix B, we discuss in detail NILSAS for discrete systems, more specifically, hyperbolic diffeomorphisms. Adjoint shadowing directions for diffeomorphisms were also defined in [32]. Because of the absence of the neutral subspace, the NILSAS algorithm for hyperbolic diffeomorphisms is easier than flows. To obtain NILSAS for diffeomorphisms, we no longer compute d_i^{wf}, $d_i^{v^*f}$, and no longer impose the second constraint in the NILSAS problem. Of course, we should change integrations to summations, and adjoint equations to their discrete counterparts.

4.2.2. Number of homogeneous adjoint solutions

Since the number of homogeneous adjoint solutions should be strictly larger than then number of unstable adjoint CLVs, which equals the number of unstable tangent CLVs, we first discuss the number of unstable CLVs, about which there are two questions: (1) whether the absolute number can be large; (2) whether the number is significantly lower than the dimension of the system. We are interested in (1) because we want to estimate the cost of NILSAS. We are interested in (2) because we want to determine whether computational efficiency can benefit from the 'non-intrusive' formulation, which restricts minimization to unstable subspaces. Roughly, the efficiency improvement due to the non-intrusive formulation is proportional to the ratio of the system dimension to unstable subspace dimension.

First, the absolute number of unstable CLVs can be large and the cost of NILSAS increases. We think maybe this is the price to pay for chaos, that is, for more chaotic systems, numerical methods should be more expensive, not only for NILSS/NILSAS, but also for other common methods such as computing long-time averages, which should take longer time to converge for more chaotic systems. Second, in a recent paper [29], based on observations on flow past a 3D cylinder, we conjectured that for open flows, CLVs active in the freestream or less turbulent regions are stable. At least for these open flows, where there are large areas of freestream, a large fraction of CLVs should be stable. For these cases, we can benefit from the non-intrusive formulation by restricting minimization to unstable subspaces.

We provide some examples on the number of unstable CLVs and the dimension of the system. Analytically, as discussed in [38], the number of unstable CLVs for general 3D turbulent flows is roughly proportional to Re^3. For numerically simulated fluids, due to numerical dissipation and turbulence models, there should be less unstable CLVs than analytic solutions. For example, for a 2D incompressible channel flow over a backward facing step at Reynolds number $Re = 2.5 \times 10^4$, there are 13 unstable CLVs in a system of dimension 4×10^4 [27]. For a 2D NACA 0012 airfoil at Mach number $Ma = 0.2$, angle of attack 20 deg, $Re = 2400$, there are less than 5 unstable CLVs for different implementations with system dimension ranging from 7×10^3 to 8×10^5 [39]. For a 3D turbulent channel flow with $Ma = 0.3$ and $Re_\tau = 180$ on a domain of size $4\pi \times 2 \times 2\pi$, there are about 1.5×10^3 unstable CLVs out of a system of dimension 2.2×10^6 [41]. For a 3D weakly turbulent flow over a cylinder at $Re = 5.2 \times 10^2$, there are 20 unstable out of 1.9×10^6 [29]. For a weather model, PUMA, there are about 65 unstable out of more than 10^5 [40]. All of the above numerical fluid problems have unstable CLVs less than 0.1% of the system dimension. On the other hand, there are cases where significant part of all CLVs are unstable, such as Hamiltonian systems. To conclude, we believe that although the cost of NILSS and NILSAS can be high, for many cases, the idea of non-intrusive formulation is still important for achieving the highest possible computational efficiency.

In our procedure list we listed M in the setting of NILSAS and required it strictly larger than m_{us}, the number of unstable CLVs. How should we know m_{us} before running NILSAS? And what if we chose initial M smaller than required? First, the number of unstable modes is roughly positively related to how chaotic the flow is. This is not a rigorous criterion but readers can look at some test cases to have a rough sense. But there is not any precise method that allows us to know the exact number at the first glance. Second, even if we started with an insufficient M, we can add adjoint solutions inductively in NILSAS, rather than recomputing everything all over again. More specifically, in the NILSAS problem in equation (26) and the sensitivity formula in equation (28), say we want to add k more adjoint solutions, then coefficients arrays $d_i^{wv^*}$, d_i^{wf}, $d_i^{wf_s}$ and b_i, should be augmented by k more entries, while the old coefficient arrays are not changed inside the new arrays; similarly, the coefficient matrices C_i, R_i should be augmented by k rows and k columns.

The headache of choosing an initial M is further relieved by the fact that adjoint solutions can be more efficiently computed in batches. Within each segment, we can accelerate NILSAS by taking advantage of the fact that all adjoint solutions, both homogeneous and inhomogeneous, use the same Jacobian f_u. If the numerical integration is vectorized, we can integrate all adjoint solutions simultaneously without repeatedly loading f_u into the computer CPU, which is the most time-consuming procedure in the numerical integration. At each time step, instead of several matrix-vector products, we can perform one matrix-matrix products, where the second matrix is composed of several adjoint solutions; then we add the right-hand-side to the inhomogeneous adjoint solution. For example, for a 4th order IEDG solver, the marginal cost for one more adjoint solution can be only, say 0.037, of the first adjoint solution. In this scenario we should start NILSAS and then add adjoint solutions by batches on the order of $1/0.037 \approx 27$ adjoint solutions per batch. This should be further

faster than adding adjoint solutions one by one.[1] However, our later implementation in this paper does not yet have this vectorized feature.

4.2.3. Other settings of NILSAS

It is required by the algorithm that we run primal system long enough before the main part of NILSAS, so that our initial condition is on the attractor. In general, we can not know very well what is 'long enough' before we do any computations, and this run-up time is determined empirically as the time when the flow field starts to repeat itself. In a typical scenario, we would run a primal simulation before taking interest in any sensitivities. When running that primal simulation, there is the same question of when we reach the stage that enough long-time behavior has been captured: typically this is indicated by that several objective functions began to oscillate around some averaged values.

We should also determine the time length T on which we run NILSAS. In practice T is determined empirically as the time when the sensitivity computed by NILSAS converges to within the uncertainty bound we desire. However, there is one caveat that the adjoint solutions are computed backwards in time. Now if we find T insufficient, we can not add time after T without recomputing all adjoint solutions, since integration adjoint solutions forward in time is very unstable. Rather, we should add time before our current trajectory. In practice, we should run our primal simulation till enough long-time behavior has been captured, then start computing adjoint solutions from the end of that primal trajectory. We have found that typically NILSAS requires a shorter trajectory to compute sensitivity than that required to reflect average behavior.

Then we discuss the choice of segment length ΔT. Similar to NILSS, ΔT is determined by that within one segment, the leading adjoint CLV does not dominate the M-th adjoint CLV. This is because otherwise we would have covariant matrices, $\{C_i\}_{i=0}^{K-1}$, with small condition number, which would lead to eventually the poor condition of the NILSAS problem in equation (26). We recommend $\Delta T(\lambda_1 - \lambda_M)$ to be $O(1)$, in which case within one segment, the leading CLV would grow to be about $e^1 = 2.7$ time larger than the M-th CLV.

A related question is that if the leading LE is large and we select a small ΔT, will the cost of frequent rescaling offset the cost reduction due to non-intrusive formulation? First, as we discussed in section 4.2.2, there are a lot of fluid systems whose CLVs are mostly stable, in which case the non-intrusive formulation is beneficial. Second, our understanding is that the numerical methods should have smaller time steps for more chaotic systems, to capture accurate motions on all scales. Hence one segment, although is shorter in physical time, may still contain many small time steps. As a result, the rescaling may not be more frequent for more chaotic systems. Again, we call for more research on numerical schemes and LE spectrum, especially for systems other than fluid or extremely chaotic systems.

4.2.4. Comparison with other shadowing algorithms

There are currently several variants of NILSS [27], such as the Finite-Difference NILSS (FD-NILSS) [28] and discrete adjoint NILSS [30]. NILSAS, as well as these NILSS variants, bears part of the merit of 'non-intrusive' formulation, that is, in comparison to LSS, the minimization problems in these algorithms are constrained to the unstable subspaces. Hence, for many real-life problems, where the unstable subspaces have significantly lower dimension than the dynamical systems, these algorithms should be significantly faster than LSS.

We compare NILSAS with variants of NILSS in Table 1. In particular, we want to compare in more detail the two adjoint algorithms: discrete adjoint NILSS versus NILSAS. NILSAS should be easier to implement than the discrete adjoint NILSS since it does not require tangent solvers, and requires less modification to existing adjoint solvers. Furthermore, unlike discrete adjoint NILSS, NILSAS does not explicitly depend on the fact that inner-products between adjoint and tangent homogeneous solutions are constants: since this property holds true only for analytic solutions but is typically false for numerical solutions, we think NILSAS should be more robust to implementations of tangent and adjoint solvers, and should typically have better convergence. We suggest more numerical comparison be done to compare the two methods.

5. Applications

5.1. Application on Lorenz 63 system

In this subsection we apply NILSAS to the Lorenz 63 system as an illustration.[2] Lorenz 63 is an ordinary differential equations system with three states $u = [x, y, z]$:

$$\frac{dx}{dt} = \sigma(y - x), \quad \frac{dy}{dt} = x(\rho - z) - y, \quad \frac{dz}{dt} = xy - \beta z. \tag{30}$$

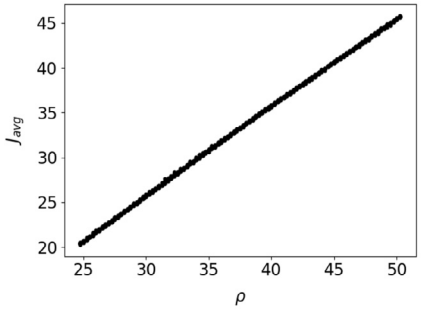
We fix $\beta = 8/3$. This system models the heat transfer in a fluid layer heated from below and cooled from above. In particular, x is the convection rate, y the horizontal temperature variation, and z the vertical temperature variation. The parameters

[1] The ideas of taking advantage of vectorized integration and the estimation on IEDG were both given during private discussion by Pablo Fernandez, who co-authored with us on finite difference NILSS (FD-NILSS), see arXiv:1711.06633.

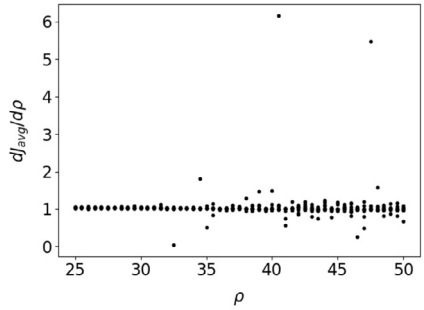
[2] The python code used for this section is at: https://github.com/niangxiu/nilsas.

Table 1
Comparison of NILSAS with NILSS, Finite Difference NILSS (FD-NILSS) and discrete adjoint (D.A) NILSS. Here 'prm', 'tan', 'adj', 'ihm' and 'hm' are short for primal, tangent, adjoint, inhomogeneous and homogeneous, respectively. M is a number strictly larger than the number of unstable CLVs. For item 4 and 5, we assume that all objectives and parameters are determined before the computation, rather than adding more objectives and parameters after the computation is done.

	NILSS	FD-NILSS	D.A. NILSS	NILSAS
1. needs prm solvers	yes	yes	yes	yes
2. needs tan solvers	yes	no	yes	no
3. needs adj solvers	no	no	yes	yes
4. cost increases with parameter numbers	yes	yes	no	no
5. cost increases with objective numbers	no	no	yes	yes
6. cost for 1 parameter and 1 objective	1 prm + 1 ihm tan + (M-1) hm tan	(M+1) prm	1 prm + 1 ihm adj + (M-1) hm tan	1 prm + 1 ihm adj + M hm adj



(a) For each value of ρ, J_{avg} is computed 20 times on randomly initialized trajectories of length 100.



(b) For each ρ, $\partial J_{avg}/\partial\rho$ is computed 10 times by NILSAS on randomly initialized trajectories of length 40.

Fig. 2. J_{avg} and $\partial J_{avg}/\partial\rho$ versus ρ for the Lorenz 63 system. Here $\sigma = 10$ is fixed.

σ and ρ are proportional to the Prandtl number and Rayleigh number. We select the instantaneous objective function as $J(u) = z$, and hence our objective J_{avg} is the averaged vertical temperature variation.

The primal system and adjoint equations are integrated via the explicit time-stepping scheme:

$$\begin{aligned} u_{k+1} &= u_k + f(u_k)\Delta t \\ \overline{w}_k &= \overline{w}_{k+1} + f_u(u_k)^T \overline{w}_{k+1}\Delta t \\ \overline{v}^*_k &= \overline{v}^*_{k+1} + f_u(u_k)^T \overline{v}^*_{k+1}\Delta t + J_u(u_k)\Delta t \end{aligned} \tag{31}$$

where the subscript k denotes the time step number in numerical integration. The time step size is $\Delta t = 0.001$. For NILSAS, time segment length is $\Delta T = 0.2$, thus there are 200 time steps per segment.

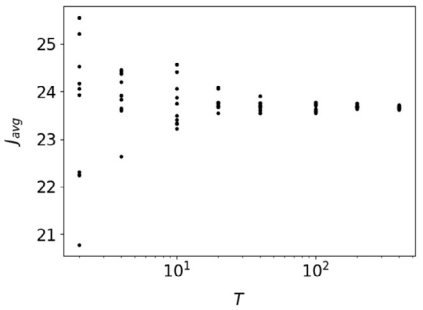
We want to determine the number of unstable CLVs for the Lorenz system. The Lyapunov exponents, $\lambda_1, \lambda_2, \lambda_3$, satisfy the following constraints [42]:

$$\lambda_1 + \lambda_2 + \lambda_3 = -(1 + \sigma + \beta) < 0\,. \tag{32}$$

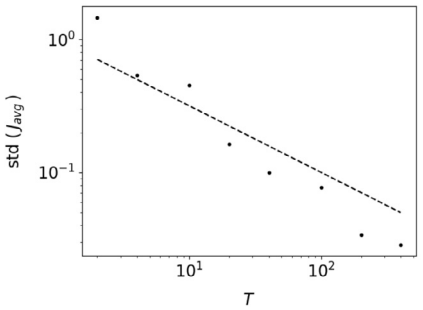
Moreover, one of these exponents corresponds to the neutral CLV so it is zero. Hence there is at most one positive exponent, so in NILSAS we set the number of homogeneous solutions $M = 2$.

We verify that NILSAS gives correct sensitivities by computing J_{avg} and $\partial J_{avg}/\partial\rho$ for different ρ, while fixing $\sigma = 10$. The Lorenz system has one quasi-hyperbolic strange attractor when $25 \le \rho < 31$, and one non-hyperbolic attractor when $31 \le \rho \le 50$: none of these cases strictly satisfies our uniform hyperbolic assumption. As shown in Fig. 2, as ρ becomes larger, the system becomes non-hyperbolic, and the sensitivity results given by NILSAS begin to oscillate. Nevertheless, NILSAS gives that $\partial J_{avg}/\partial\rho$ is approximately 1 for all ρ, which matches the trend between J_{avg} and ρ: this again shows that NILSAS can be effective for systems not satisfying assumptions of Theorem 1, as we discussed in section 4.2.1.

Then we show that both J_{avg} and the sensitivities computed by NILSAS converge as the trajectory length T gets larger, while fixing $\sigma = 10$ and $\rho = 28$. Fig. 3 shows that the standard deviation of J_{avg} reduces at the rate of $T^{-0.5}$. Fig. 4 shows that the sensitivities computed by NILSAS, with respect to both ρ and σ, converge faster than the rate of $T^{-0.5}$.

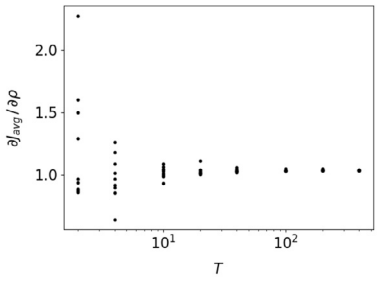


(a) For each value of trajectory length T, J_{avg} is computed 10 times on randomly initialized trajectories.

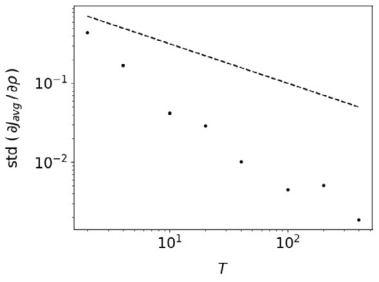


(b) The sample standard deviation of the 10 J_{avg}'s computed at each T. The dashed line is $T^{-0.5}$.

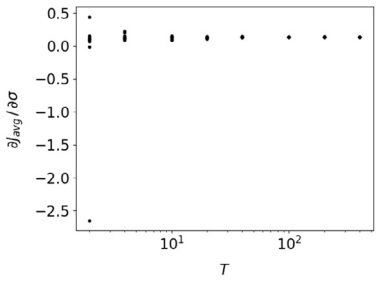
Fig. 3. Convergence of the averaged objective J_{avg} with respect to the trajectory length T. Here $\rho = 28$ and $\sigma = 10$ are fixed.



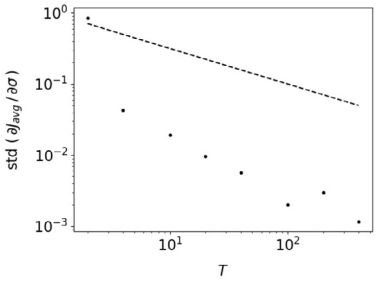
(a) For each value of T, $\partial J_{avg}/\partial\rho$ is computed by NILSAS 10 times on randomly initialized trajectories.



(b) The sample standard deviation of the 10 $\partial J_{avg}/\partial\rho$'s computed at each T. The dashed line is $T^{-0.5}$.



(c) For each value of T, $\partial J_{avg}/\partial\sigma$ is computed by NILSAS 10 times on randomly initialized trajectories.



(d) The sample standard deviation of the 10 $\partial J_{avg}/\partial\sigma$'s computed at each T. The dashed line is $T^{-0.5}$.

Fig. 4. Convergence of sensitivities computed by NILSAS with respect to the trajectory length T. Here $\rho = 28$ and $\sigma = 10$ are fixed.

NILSAS computes sensitivities with respect to multiple parameters with almost no additional cost, since the adjoint shadowing solution $\overline{v}$ does not depend on the choice of parameters. Fig. 5 illustrates the contour of J_{avg} with respect to ρ and σ, and the gradient, $[\partial J_{avg}/\partial\rho, \partial J_{avg}/\partial\sigma]$, is computed by NILSAS. Since we use the same length unit for both parameters, gradients should be perpendicular to the level sets of the objective: this is indeed the case, and it shows NILSAS gives correct gradient information.

Finally, we draw the norm of an adjoint shadowing direction in Fig. 6. As we can see from the left plot, the norm of the adjoint shadowing direction does not grow exponentially, satisfying the third property of adjoint shadowing directions. Moreover, as shown in the right plot, the adjoint shadowing direction computed by NILSAS is continuous. This shows that our dividing trajectory technique indeed allows us to recover a continuous adjoint shadowing direction.

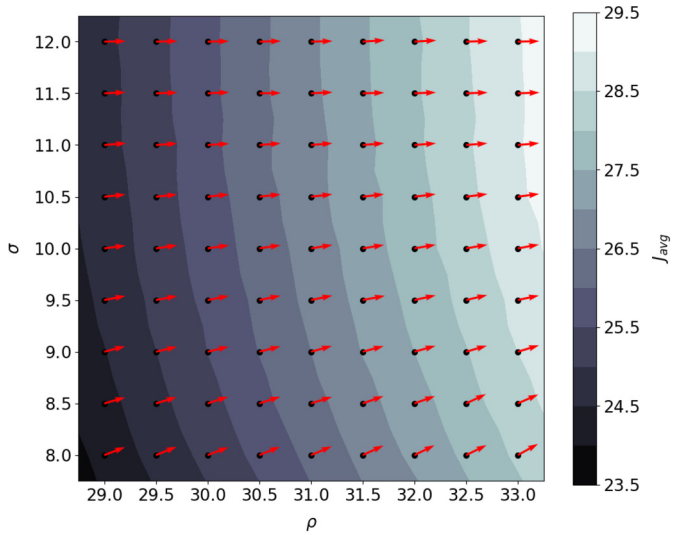


Fig. 5. Gradients computed by NILSAS. The contour is of J_{avg} with respect to ρ and σ, and arrows are gradient vectors. Here J_{avg}'s are averaged over 20 randomly initialized trajectories of length 100, while gradients computed by NILSAS are averaged over 10 randomly initialized trajectories of length 40. The arrow length is 0.2 times the gradient norm. NILSAS computes one gradient, composed of two sensitivities to two parameters, in one run.

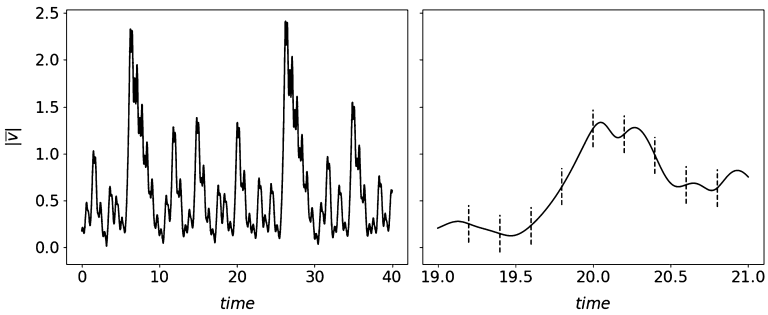


Fig. 6. Norm of the adjoint shadowing direction computed by NILSAS for the Lorenz system, with $\rho = 28$ and $\sigma = 10$. Left: plot on the entire trajectory time span. Right: zoom onto time span from 19 to 21. The vertical dashed lines marks different time segments.

5.2. Application on a turbulent flow past a three-dimensional cylinder

In this subsection, we apply NILSAS to a 3D subsonic flow over a cylinder at Reynolds number $Re = 1100$ and Mach number $Ma = 0.093$.[3] The flow-wise length of the domain is $60d$, where $d = 0.25$ mm is the diameter of the cylinder. The Reynolds number is defined using the diameter of the cylinder and the density, velocity and viscosity of inflow. The span-wise extent, at $z = 2d$, is sufficient to capture most of the important flow features, like a turbulent wake and flow separation. The front view of our fluid problem is shown in Fig. 7.

We use compressible Navier-Stokes equations with the ideal gas law approximating the thermodynamic state equation [43]. The gas is assumed to be air. More specifically, the governing equations are:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0\,,$$
$$\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) + \nabla p = \nabla \cdot \sigma +\,,$$
$$\frac{\partial (\rho E)}{\partial t} + \nabla \cdot (\rho E \mathbf{u} + p\mathbf{u}) = \nabla \cdot (\mathbf{u} \cdot \sigma + \alpha \rho \gamma \nabla e)\,,$$
$$\sigma = \mu(\nabla \mathbf{u} + \nabla \mathbf{u}^T) - \frac{2\mu}{3}(\nabla \cdot \mathbf{u})\mathbf{I}\,, \quad c = \sqrt{\frac{\gamma p}{\rho}}\,,$$

[3] The flow solver, adFVM, used for this section is at: https://github.com/chaitan3/adFVM, the particular file that implements the NILSAS algorithm used in this case is apps/nilsas.py.

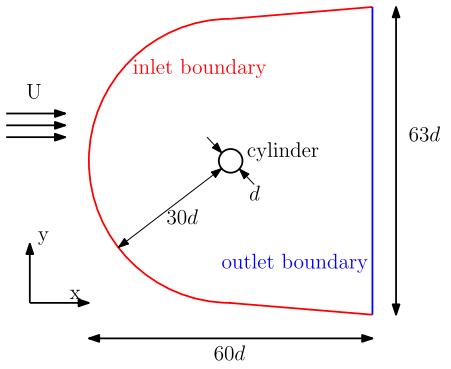


Fig. 7. Geometry used in the simulation of a 3D flow past a cylinder. The span-wise extent of the computational domain is 2d.

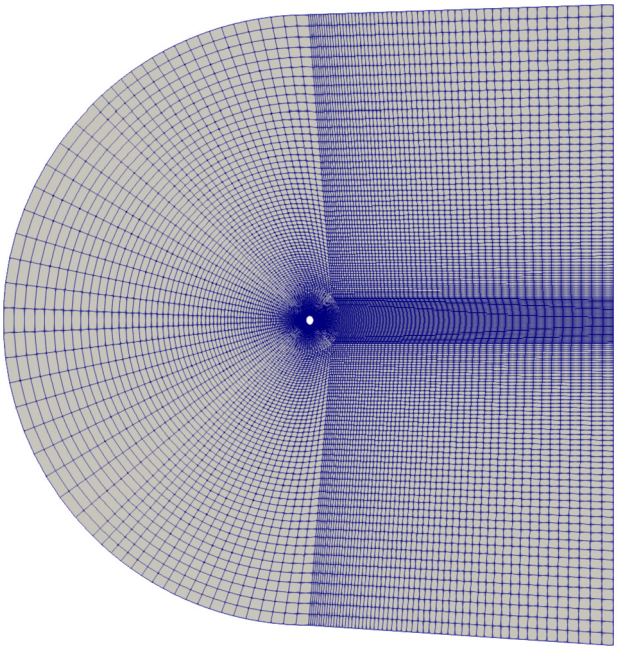


Fig. 8. Front view of the mesh for the flow over cylinder problem. This is an unstructured hexahedral mesh with approximately 7×10^5 cells, with 50 cells in the span-wise direction.

$$p = (\gamma - 1)\rho e\,, \quad e = E - \frac{\mathbf{u}\cdot\mathbf{u}}{2}\,. \tag{33}$$

Here ρ is the density, $\mathbf{u}$ is the velocity vector, ρE is the total energy, p is pressure, e is internal energy of the fluid, c is the speed of sound, $\gamma = 1.4$ is the isentropic expansion factor and μ is the viscosity field modeled using Sutherland's law for air

$$\mu = \frac{C_s T^{3/2}}{T + T_s} \tag{34}$$

where $T_s = 110.4$ K and $C_s = 1.458 \times 10^{-6}$ kg/m s$\sqrt{K}$. α is the thermal diffusivity modeled using

$$\alpha = \frac{\mu}{\rho Pr} \tag{35}$$

where $Pr = 0.71$ is the Prandtl number.

We use an unstructured hexahedral mesh with approximately 7×10^5 cells, with 50 cells in the span-wise direction. The front view of our mesh is shown in Fig. 8.

We use a second order finite volume method (FVM) [44] for unstructured hexahedral meshes. The central differencing scheme is used to interpolate cell averages of the flow solution onto faces of the mesh [45]. The numerical fluxes for the conservative flow variables are computed using the Roe approximate Riemann solver [46]. An explicit time integration

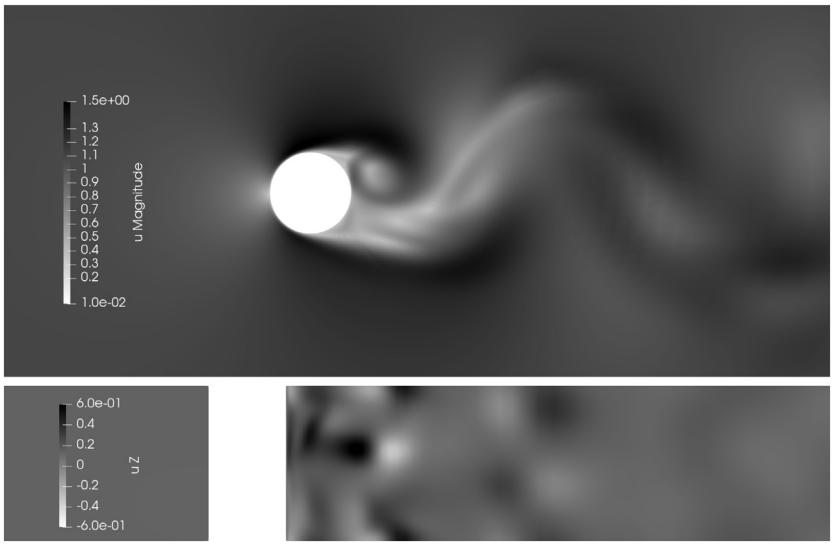


Fig. 9. Instantaneous visualization of the flow field. Top: vertical cross-section, plotted by the magnitude of velocity. Bottom: horizontal cross-section, plotted by the span-wise velocity. The bottom picture shows the flow is 3D. All velocities are normalized by the reference velocity u_r.

scheme, the strong stability preserving third order Runge-Kutta method [47], is used for time marching the numerical flow solution. The size of the time step is determined using the acoustic Courant-Friedrichs-Lewy (CFL) condition [48], and we choose our CFL number to be 1.2. The flow solver is implemented in Python using the adFVM [49] library, which provides a high-level abstract application programming interface for writing efficient CFD applications. The flow solver is parallelized using the Message Passing Interface (MPI) library.

We use implicit Large Eddy Simulations (LES) in our numerical simulation. In an LES, the large scale eddies of the flow are resolved by the grid, while the contribution from the small scale eddies to the filtered Navier-Stokes equations are modeled using a sub-grid scale Reynolds stress model [43]. In this paper, the numerical error of the discretization scheme serves as the LES model. It has been shown that when using a relatively dissipative discretization method, the numerical viscosity from the grid can be of the same order of magnitude as the sub-grid scale viscosity [50,51], and thus can be regarded as an implicit LES model.

On the inlet boundary, we specify stagnation pressure and temperature, corresponding to a fixed Reynolds number $Re = 1100$ and a Mach number which we choose to be the system parameter. For the base case, we choose Mach number $Ma = 0.093$. Periodic boundary condition is used in the span-wise direction. The surface of the cylinder is maintained at a constant temperature of 300 K. Static pressure of 1 atmosphere unit is prescribed on the outlet boundary.

A snapshot of the flow field simulated with above settings is shown in Fig. 9. As we can see, this flow exhibits weak turbulence in the wake. In particular, the top view shows that this flow is 3D.

We choose our system parameter as the Mach number of the incoming flow. The objective function is the time-averaged normalized drag over the cylinder. More specifically,

$$J_{avg} = \frac{2}{\rho_r u_r^2 z d} \lim_{T\to\infty} \frac{1}{T} \int_0^T \int (p n_x - \mu \boldsymbol{n} \cdot \nabla u_x)\, dS\, dt. \tag{36}$$

Here the second integral is over the surface of the cylinder; $u_r = 31.4$ m/s and $\rho_r = 1.3$ kg/m^3 are the reference velocity and density of the base case, where $Ma = 0.093$. For the base case, the normalized drag and the drag coefficient are the same, whereas for other Mach numbers they are different.

We run the flow simulation for 10^6 time steps, which corresponds to approximately $720t_r$. Here the time unit t_r is the amount of time that the flow takes to traverse the length of the cylinder, that is, $t_r = d/u_r$. This time interval is sufficient to obtain a statistically converged estimate of the design objective. The standard deviation of the time-averaged objective is computed using the autoregressive time series analysis techniques described in [52] and [53], and we use one standard deviation as the confidence interval. The normalized drag for the base case is 1.2 ± 0.03. Our results reasonably matches the results from experiments [54–56], which is approximately 1.0 ± 0.15. Fig. 10 shows different objectives for different incoming Mach number.

We first estimate the sensitivity by the linear least-squares regression method using 5 data points with different parameter values. To use this linear regression method, we need to make the assumption that the relation between parameters and objectives is linear. We select one standard deviation of the relevant estimator as our confidence interval. Note that one shortcoming of the linear regression method is that the linear assumption may not be true when parameters are spaced far

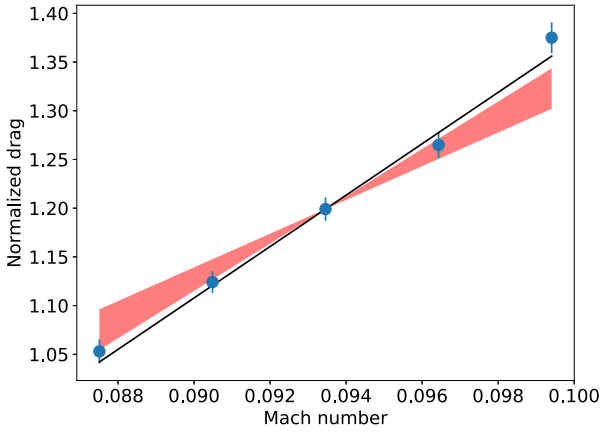


Fig. 10. Normalized drag as a function of inlet Mach number. Blue bars denote the confidence interval of the averaged normalized drag. The black line denotes the sensitivity estimated using linear regression. The red shaded region denotes the confidence interval of the sensitivity estimated using NILSAS.

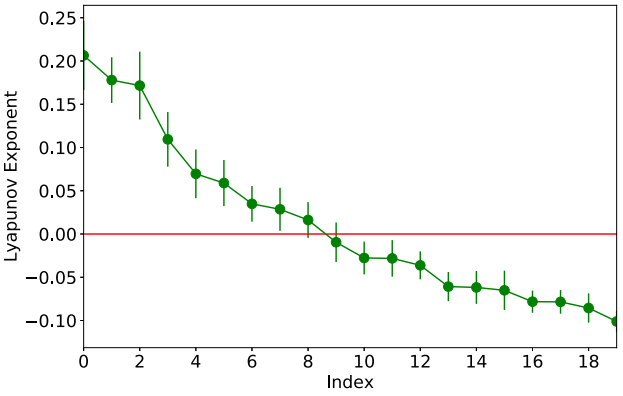


Fig. 11. Spectrum of the first 20 adjoint Lyapunov Exponents (LE). The time unit for LEs is t_r^{-1}. The largest LE is $0.21t_r^{-1}$, meaning in one time unit t_r, the norm of the first adjoint CLV becomes $e^{0.21} = 1.23$ times larger.

apart such that a linear approximation no longer holds on the dataset; on the other hand, when parameters are too close, the uncertainty in the objectives will lead to large error in the sensitivity. In the base case, the sensitivity of drag with respect to the inflow Mach number, given by linear regression, is 25.0 ± 2.1. This sensitivity is visualized in Fig. 10.

Adjoint LEs are shown in Fig. 11, where the confidence interval is also selected as one standard deviation given by autoregressive time series analysis. The subsonic flow over a 3D cylinder has $m_{us} = 9$ unstable adjoint CLVs. In NILSAS, the number of homogeneous adjoint solutions computed is set to $M = 20$. We suggest more research be done to investigate how the number of unstable CLVs grow as the flow becomes more turbulent, not only for this particular open flow problem, but for wall-bounded flows as well.

The number of segments in NILSAS is $K = 100$ and the number of time steps per segment is 500. Each segment roughly corresponds to $0.4t_r$. Consequently, the time length of trajectory used in NILSAS is $40t_r$, which is much lower than that required to obtain a reasonably accurate sensitivity using the linear regression method. In this particular implementation, corresponding to the discussion in section 4.2.1, on segment $[t_i, t_{i+1}]$, we approximate integrations in equation (23) using snapshots at t_{i+1}. As a result, we have $C_i = I$, $d_i^{wv^*} = 0$: this approximation eases the implementation responsible for storing adjoint solutions.

The sensitivity computed by NILSAS is 20.8 ± 3.5, which costs about a week on a computer with 64 cores, is visualized in Fig. 10. Here the confidence interval is also selected as one standard deviation given by autoregressive time series analysis. Comparing to the sensitivity estimated via linear regression methods, the relative difference is less than 20%. As we can see, the sensitivity computed by NILSAS correctly reflects the trend between parameters and objectives.

We remark that NILSAS may work for systems do not strictly satisfy the assumptions in Theorem 1, and our fluid problem is such an example. First, our system has at least two neutral CLVs: the first one corresponds to the common time translation of continuous dynamical systems, and the second corresponds to span-wise translations due to the periodic boundary conditions. Second, due to the similarity of this fluid problem with the one investigated in [29], whose tangent CLVs appear to have occasional tangencies, it is reasonable to assume that adjoint CLVs in our current system may also have

occasional tangencies. Still, like NILSS did in [29], NILSAS computes a correct sensitivity: this encourages us to test NILSS and NILSAS on more general chaotic systems.

We compare computational cost for sensitivity analysis via the linear regression method and NILSAS. The linear regression method runs the primal solver for a total of 5×10^6 steps. NILSAS runs the primal solver and 21 adjoint solvers, each for 5×10^4 steps, which leads to 1.1×10^6 steps in total. To build more favor towards NILSAS, note that, first, adjoint solvers can be further accelerated due to the vectorization we discussed in section 4.2.2; second, NILSAS has no additional cost for sensitivities to multiple parameters. For chaotic problems with a higher number of positive LEs, the cost of NILSAS increases; however, if the percentage of positive LEs is still low, the non-intrusive formulation can still be a key technique for designing fast sensitivity algorithms.

6. Conclusions

To compute the gradient of long-time averaged objectives in chaotic systems, we develop the Non-Intrusive Least Squares Adjoint Shadowing (NILSAS) algorithm, which approximates the adjoint shadowing direction by a 'non-intrusive' formulation, which is a least squares problem in the unstable adjoint subspace. NILSAS is demonstrated on the Lorenz 63 system and a turbulent 3D flow over a cylinder, where it gives accurate sensitivities for both cases.

Similar to NILSS [27], NILSAS can be implemented with little modification to existing adjoint solvers, and its minimization is carried out only in the unstable adjoint subspace. Unlike NILSS, NILSAS has the benefit of adjoint approaches that its cost does not increase with the number of parameters; thus making NILSAS ideal for applications where there are many parameters, or where f_s is unknown a priori. NILSAS does not require tangent solvers, and is easy to implement.

Appendix A. Solving the NILSAS problem

We discuss one way to solve the NILSAS problem in equation (26). The corresponding Lagrange function is:

$$\begin{aligned}&\sum_{i=0}^{K-1} \frac{1}{2}(a_i)^T C_i a_i + (d_i^{wv^*})^T a_i \\ &+ \sum_{i=1}^{K-1} \lambda_i^T \left(a_{i-1} - R_i a_i - b_i\right) + \lambda' \left(\sum_{i=0}^{K-1} (d_i^{wf})^T a_i + \sum_{i=0}^{K-1} d_i^{v^* f} \right),\end{aligned} \tag{A.1}$$

where λ_i is the Lagrange multiplier for the continuity condition at t_i. By the Lagrange multiplier method, the minimizer for the NILSAS problem is at the solution of the following linear equation systems:

$$\begin{bmatrix} C & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} a \\ \lambda \end{bmatrix} = \begin{bmatrix} -d \\ b \end{bmatrix}, \tag{A.2}$$

where the block matrices $C \in \mathbb{R}^{MK \times MK}$, $B \in \mathbb{R}^{(MK-M+1)\times MK}$, vectors $a, d \in \mathbb{R}^{MK}$ and $\lambda, b \in \mathbb{R}^{MK-M+1}$. More specifically,

$$\begin{aligned} C &= \begin{bmatrix} C_0 & & & \\ & C_1 & & \\ & & \ddots & \\ & & & C_{K-1} \end{bmatrix}, \quad B = \begin{bmatrix} I & -R_1 & & & \\ & I & -R_2 & & \\ & & \ddots & \ddots & \\ & & & I & -R_{K-1} \\ (d_0^{wf})^T & & \cdots & & (d_{K-1}^{wf})^T \end{bmatrix}, \\ a &= \begin{bmatrix} a_0 \\ \vdots \\ a_{K-1} \end{bmatrix}, \quad \lambda = \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_{K-1} \\ \lambda' \end{bmatrix}, \quad d = \begin{bmatrix} d_0^{wv^*} \\ \vdots \\ d_{K-1}^{wv^*} \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_{K-1} \\ -\sum_{i=0}^{K-1} d_i^{v^* f} \end{bmatrix}, \end{aligned} \tag{A.3}$$

where $\{C_i\}_{i=0}^{K-1}$, $\{R_i\}_{i=1}^{K-1} \subset \mathbb{R}^{M\times M}$; $\{a_i\}_{i=0}^{K-1}$, $\{d_i^{wf}\}_{i=0}^{K-1}$, $\{d_i^{wv^*}\}_{i=0}^{K-1}$, $\{\lambda_i\}_{i=1}^{K-1}$, $\{b_i\}_{i=1}^{K-1} \subset \mathbb{R}^M$; λ', $\{d_i^{v^* f}\}_{i=0}^{K-1} \subset \mathbb{R}$.

We can solve the Schur complement of equation (A.2) for λ:

$$-BC^{-1}B^T\lambda = BC^{-1}d + b, \tag{A.4}$$

where C^{-1} can be computed via inverting each diagonal block in C. Then we can compute a by:

$$a = -C^{-1}(B^T\lambda + d). \tag{A.5}$$

Appendix B. NILSAS on discrete systems

B.1. Backgrounds and notations

We provide a brief introduction on discrete dynamical systems, in particular, hyperbolic diffeomorphisms. More details are provided in [32]. First, the governing equation for a discrete dynamical system is:

$$u_{l+1} = f(u_l, s), \quad l \geq 0\,. \tag{B.1}$$

The objective is:

$$J_{avg} := \lim_{N\to\infty} \frac{1}{N} \sum_{l=0}^{N-1} J(u_l, s). \tag{B.2}$$

Similar to flows, we assume $u_l \in \mathbb{R}^m$, and $f(u, s)$ and $J(u, s)$ are smooth. We also assume f is a diffeomorphism in u, that is, for each fixed s, f has a smooth inverse. Also, for simplicity of notations, we assume there is only one parameter $s \in \mathbb{R}$.

We first look at the tangent equations. The homogeneous tangent diffeomorphism is:

$$w_{l+1} = f_{ul} w_l\,, \tag{B.3}$$

where the second subscript of f_{ul} indicate where the partial derivative is evaluated, that is, $f_{ul} := \partial f/\partial u(u_l, s)$. A tangent CLV with exponent λ is a homogeneous tangent solution $\{\zeta_l\}_{i=0}^{\infty}$ such that there is constant C, for any integer l_1, l_2, $\|\zeta_{l_2}\| \leq Ce^{\lambda(l_2-l_1)}\|\zeta_{l_1}\|$. The uniform hyperbolicity for diffeomorphisms is defined as that all LEs are not 1.

On the adjoint side, the homogeneous adjoint diffeomorphism is defined as:

$$\overline{w}_l = f_{ul}^T \overline{w}_{l+1}\,, \tag{B.4}$$

where $\cdot^T$ is the matrix transpose. The particular inhomogeneous adjoint diffeomorphism we will be using is:

$$\overline{v}_l = f_{ul}^T \overline{v}_{l+1} + J_{ul}. \tag{B.5}$$

On a trajectory $\{u_l\}_{l=0}^{\infty}$ on the attractor, the adjoint shadowing direction $\{\overline{v}_l\}_{l=0}^{\infty}$ is a sequence with the following properties:

1. $\{\overline{v}_l\}_{l=0}^{\infty}$ solves an inhomogeneous adjoint equation:

$$\overline{v}_l = f_{ul}^T \overline{v}_{l+1} + J_{ul}\,, \tag{B.6}$$

2. $\overline{v}_0$ has zero component in the unstable adjoint subspace,
3. $\|\overline{v}_l\|$ is bounded by a constant for all $l \geq 0$.

It was proved in [32] that for a uniform hyperbolic diffeomorphism with a global compact attractor, on a trajectory on the attractor, there exists a unique adjoint shadowing direction. Further, we have the adjoint sensitivity formula:

$$\frac{dJ_{avg}}{ds} = \lim_{N\to\infty} \frac{1}{N} \sum_{l=0}^{N-1} \left(\langle \overline{v}_{l+1}, f_{sl}\rangle + J_{sl}\right)\,. \tag{B.7}$$

B.2. Procedure list of NILSAS

We provide a procedure list for the NILSAS algorithm on discrete chaotic systems, more specifically, hyperbolic diffeomorphisms. To start with, we need an inhomogeneous adjoint solver and a homogeneous adjoint solver, both can take arbitrary terminal conditions. The inhomogeneous adjoint equation still has right-hand-side $-J_u$, same as many existing adjoint solvers for discrete systems. We provide the following data: 1) the number of homogeneous adjoint solutions, $M \geq m_{us}$, where m_{us} is the number of unstable CLVs, note that because the lack of neutral CLV, we can use one less homogeneous adjoint solution; 2) the total number of segments, K; 3) number of steps in one segment, L.

We can have three subscripts, the first, typically being u or s, indicates this term is a partial derivative; the second, typically being i, 0, or K, indicates the segment number; the third, typically being l, 0, or L, indicates the step number inside a segment. Disappearance of the first subscript means that term is not a partial derivative. Disappearance of the third subscript means either we are considering all steps in a segment, or that term is defined only once per segment interface.

1. Integrate the primal system for sufficiently many steps so that the initial condition, u_{00}, is on the attractor.
2. Compute the trajectory u_{il} for $0 \leq i \leq K-1$ and $0 \leq l \leq L$. Here we assume the step at end of each segment overlaps with the start of next segment, that is, $u_{iL} = u_{i+1,0}$.

3. Generate terminal conditions for $\overline{W}_i$ and $\overline{v}_i^*$ on the last segment $i = K-1$:
   (a) Randomly generate a $m \times M$ full rank matrix, Q'. Perform QR factorization: $Q_K R_K = Q'$.
   (b) Set $p_K = 0$.
4. Compute $\overline{W}_i$ and v_i^* on all segments. For $i = K-1$ to $i = 0$ do:
   (a) To get $\overline{W}_{il}$, whose columns are homogeneous adjoint solutions on segment i, solve:

$$\overline{w}_{il} = f_{uil}^T \overline{w}_{i,l+1}\,, \quad \overline{W}_{iL} = Q_{i+1}\,. \tag{B.8}$$

   To get $\overline{v}_i^*(t)$, solve the inhomogeneous adjoint equation:

$$\overline{v}_{il} = f_{uil}^T \overline{v}_{i,l+1} + J_{uil}\,, \quad \overline{v}_{iL}^* = p_{i+1}\,. \tag{B.9}$$

   (b) Compute the following integrations.

$$\begin{aligned} &C_i = \sum_{l=1}^{L} \overline{W}_{il}^T \overline{W}_{il} dt\,, \quad d_i^{wv^*} = \sum_{l=1}^{L} \overline{W}_{il}^T \overline{v}_{il}^* dt\,, \quad d_i^{J_s} = \sum_{l=1}^{L} J_{sil} dt\,, \\ &d_i^{wf_s} = \sum_{l=1}^{L} \overline{W}_{il}^T f_{si,l-1} dt\,, \quad d_i^{v^* f_s} = \sum_{l=1}^{L} \overline{v}_{il}^{*T} f_{si,l-1} dt\,, \end{aligned} \tag{B.10}$$

   where $d_i^{wv^*}$, $d_i^{wf_s} \in \mathbb{R}^M$; $d_i^{v^* f_s}$, $d_i^{J_s} \in \mathbb{R}$; $C_i \in \mathbb{R}^{M \times M}$ is the covariant matrix. Note that when multiplying adjoint solutions with f_s, their time steps are not the same: this asymmetry is the same as that in equation (B.7). We are not sure yet if this technical detail can be neglected in practice.
   (c) Orthonormalize homogeneous adjoint solutions via QR factorization:

$$Q_i R_i = \overline{W}_{i0} \tag{B.11}$$

   (d) Rescale the inhomogeneous adjoint solution using Q_i:

$$p_i = \overline{v}_{i0}^* - Q_i b_i\,, \quad \text{where } b_i = Q_i^T \overline{v}_{i0}^*\,. \tag{B.12}$$

5. Compute the adjoint shadowing direction $\{\overline{v}_{il}\}$ for $0 \le i \le K-1$ and $0 \le l \le L$.
   (a) Solve the NILSAS problem on multiple segments:

$$\begin{aligned} \min_{a_0, \cdots, a_{K-1} \in \mathbb{R}^M} &\sum_{i=0}^{K-1} \frac{1}{2} (a_i)^T C_i a_i + (d_i^{wv^*})^T a_i, \quad \text{s.t.} \\ &a_{i-1} = R_i a_i + b_i\,, \quad i = 1, \cdots, K-1\,. \end{aligned} \tag{B.13}$$

   This is a least squares problem in $\{a_i\}_{i=0}^{K-1} \subset \mathbb{R}^M$. Note we do not have the other constraint as NILSAS in the continuous case.
   (b) On each time segment i, $\overline{v}_{il}$ is given by

$$\overline{v}_{il} = \overline{v}_{il}^* + \overline{W}_{il} a_i. \tag{B.14}$$

6. Compute the derivative by:

$$\frac{dJ_{avg}}{ds} \approx \frac{1}{KL} \sum_{i=0}^{K-1} \left(d_i^{v^* f_s} + a_i^T d_i^{wf_s} + d_i^{J_s} \right) \tag{B.15}$$

References

[1] J.J. Reuther, A. Jameson, J.J. Alonso, M.J. Rimllnger, D. Saunders, Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, Part 2, J. Aircr. 36 (1999) 61–74.
[2] A. Jameson, Aerodynamic design via control theory, J. Sci. Comput. 3 (1988) 233–260.
[3] T.R. Bewley, Flow control: new challenges for a new Renaissance, Prog. Aerosp. Sci. 37 (2001) 21–58.
[4] T.R. Bewley, P. Moin, R. Temam, DNS-based predictive control of turbulence: an optimal benchmark for feedback algorithms, J. Fluid Mech. 447 (2001) 179–225.
[5] J. Tromp, C. Tape, Q. Liu, Seismic tomography, adjoint methods, time reversal and banana-doughnut kernels, Geophys. J. Int. 160 (2005) 195–216.
[6] R. Becker, R. Rannacher, An optimal control approach to a posteriori error estimation in finite element methods, Acta Numer. 10 (2001) 1–102.
[7] M.B. Giles, E. Süli, Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality, Acta Numer. 11 (2002) 145–236.
[8] K.J. Fidkowski, D.L. Darmofal, Review of output-based error estimation and mesh adaptation in computational fluid dynamics, AIAA J. 49 (2011) 673–694.
[9] P.-O. Persson, J. Peraire, Curved mesh generation and mesh refinement using Lagrangian solid mechanics, in: 47th AIAA Aerospace Sciences Meeting (AIAA 2009-949), Aerospace Sciences Meetings, American Institute of Aeronautics and Astronautics, 2009, pp. 1–11.

[10] J.-N. Thepaut, P. Courtier, Four-dimensional variational data assimilation using the adjoint of a multilevel primitive-equation model, Q. J. R. Meteorol. Soc. 117 (1991) 1225–1254.
[11] P. Courtier, J. Derber, R. Errico, J. Louis, T. Vukićević, Important literature on the use of adjoint, variational methods and the Kalman filter in meteorology, Tellus A 45 (1993) 342–357.
[12] Y.M. Marzouk, K. Willcox, Uncertainty quantification, in: N.J. Higham (Ed.), The Princeton Companion to Applied Mathematics, Princeton University Press, 2015, pp. 131–133.
[13] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[14] A. Ni, C. Talnikar, Linear Range in Gradient Descent, arXiv:1905.04561, 2019.
[15] D.J. Lea, M.R. Allen, T.W.N. Haine, Sensitivity analysis of the climate of a chaotic system, Tellus, Ser. A Dyn. Meteorol. Oceanogr. 52 (2000) 523–532.
[16] G.L. Eyink, T.W.N. Haine, D.J. Lea, Ruelle's linear response formula, ensemble adjoint schemes and Lévy flights, Nonlinearity 17 (2004) 1867–1889.
[17] R.V. Abramov, A.J. Majda, Blended response algorithms for linear fluctuation-dissipation for complex nonlinear dynamical systems, Nonlinearity 20 (2007) 2793–2821.
[18] R.V. Abramov, A.J. Majda, New approximations and tests of linear fluctuation-response for chaotic nonlinear forced-dissipative dynamical systems, J. Nonlinear Sci. 18 (2008) 303–341.
[19] V. Lucarini, F. Ragone, F. Lunkeit, Predicting climate change using response theory: global averages and spatial patterns, J. Stat. Phys. 166 (2017) 1036–1064.
[20] A. Gritsun, V. Lucarini, Fluctuations, response, and resonances in a simple atmospheric model, Phys. D: Nonlinear Phenom. 349 (2017) 62–76.
[21] R. Bowen, Markov partitions for axiom a diffeomorphisms, Am. J. Math. 92 (1970) 725–747.
[22] S.Y. Pilyugin, Shadowing in structurally stable flows, J. Differ. Equ. 140 (1997) 238–265.
[23] Q. Wang, Convergence of the least squares shadowing method for computing derivative of ergodic averages, SIAM J. Numer. Anal. 52 (2014) 156–170.
[24] Q. Wang, R. Hu, P. Blonigan, Least squares shadowing sensitivity analysis of chaotic limit cycle oscillations, J. Comput. Phys. 267 (2014) 210–224.
[25] M. Chater, A. Ni, P.J. Blonigan, Q. Wang, Least squares shadowing method for sensitivity analysis of differential equations, SIAM J. Numer. Anal. 55 (2017) 3030–3046.
[26] A. Ni, P.J. Blonigan, M. Chater, Q. Wang, Z. Zhang, Sensitivity analysis on chaotic dynamical system by Non-Intrusive Least Square Shadowing (NI-LSS), in: 46th AIAA Fluid Dynamics Conference, AIAA AVIATION Forum, AIAA 2016-4399, American Institute of Aeronautics and Astronautics, 2016, pp. 1–16.
[27] A. Ni, Q. Wang, Sensitivity analysis on chaotic dynamical systems by Non-Intrusive Least Squares Shadowing (NILSS), J. Comput. Phys. 347 (2017) 56–77.
[28] A. Ni, Q. Wang, P. Fernandez, C. Talnikar, Sensitivity analysis on chaotic dynamical systems by Finite Difference Non-Intrusive Least Squares Shadowing (FD-NILSS), arXiv:1711.06633, 2018, pp. 1–21.
[29] A. Ni, Hyperbolicity, shadowing directions and sensitivity analysis of a turbulent three-dimensional flow, J. Fluid Mech. 863 (2019) 644–669.
[30] P.J. Blonigan, Adjoint sensitivity analysis of chaotic dynamical systems with non-intrusive least squares shadowing, J. Comput. Phys. 348 (2017) 803–826.
[31] N. Chandramoorthy, Q. Wang, L. Magri, S.H.K. Narayanan, P. Hovland, A. Ni, Algorithmic differentiation of shadowing sensitivities in chaotic systems, in: SIAM Workshop on Combinatorial Scientific Computing, Bergen, Norway, pp. 1–18.
[32] A. Ni, Adjoint shadowing directions in hyperbolic systems for sensitivity analysis, arXiv:1807.05568, 2018, pp. 1–23.
[33] P.V. Kuptsov, U. Parlitz, Theory and computation of covariant Lyapunov vectors, J. Nonlinear Sci. 22 (2012) 727–762.
[34] A.B. Katok, B.A. Hasselblatt, Introduction to the Modern Theory of Dynamical Systems, Encyclopedia of Mathematics and Its Applications, vol. 54, Cambridge University Press, 1997.
[35] F. Ginelli, H. Chaté, R. Livi, A. Politi, Covariant Lyapunov vectors, J. Phys. A, Math. Theor. 46 (2013) 254005.
[36] G. Gallavotti, E.G.D. Cohen, Dynamical ensembles in stationary states, J. Stat. Phys. 80 (1995) 931–970.
[37] G. Gallavotti, Chaotic hypothesis, Scholarpedia 3 (1) (2008) 5906.
[38] P. Constantin, C. Foias, O.P. Manley, R. Temam, Determining modes and fractal dimension of turbulent flows, J. Fluid Mech. 150 (1985) 427–440.
[39] P. Fernandez, Q. Wang, Lyapunov spectrum of the separated flow around the NACA 0012 airfoil and its dependence on numerical discretization, J. Comput. Phys. 350 (2017) 453–469.
[40] L. De Cruz, S. Schubert, J. Demaeyer, V. Lucarini, S. Vannitsem, Exploring the Lyapunov instability properties of high-dimensional atmospheric and climate models, Nonlinear Process. Geophys. 25 (2018) 387–412.
[41] P. Blonigan, P. Fernandez, S. Murman, Q. Wang, G. Rigas, L. Magri, Toward a chaotic adjoint for LES, in: Center for Turbulence Research, Proceedings of the Summer Program, Stanford, pp. 385–394.
[42] J. Bovy, Lyapunov Exponents and Strange Attractors in Discrete and Continuous Dynamical Systems, Technical Report, KU Leuven University, 2004, Theoretical Physics Project.
[43] E. Garnier, N. Adams, P. Sagaut, Large Eddy Simulation for Compressible Flows, Scientific Computation, Springer, 2009.
[44] R.J. LeVeque, Finite Volume Methods for Hyperbolic Problems, Cambridge University Press, 2002.
[45] H.K. Versteeg, W. Malaskekera, An Introduction to Computational Fluid Dynamics: The Finite Volume Method, Pearson, 1995.
[46] P. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, J. Comput. Phys. 43 (1981) 357–372.
[47] C.B. Macdonald, Constructing High-Order Runge-Kutta Methods With Embedded Strong-Stability-Preserving Pairs, Ph.D. thesis, Simon Fraser University, 2003.
[48] R. Courant, K. Friedrichs, H. Lewy, On the partial difference equations of mathematical physics, IBM J. Res. Dev. 11 (1967) 215–234.
[49] C. Talnikar, Q. Wang, A two-level computational graph method for the adjoint of a finite volume based compressible unsteady flow solver, Parallel Comput. 81 (2019) 68–84.
[50] C.-H. Moeng, J.C. Wyngaard, Evaluation of turbulent transport and dissipation closures in second-order modeling, J. Atmos. Sci. 46 (1989) 2311–2330.
[51] P. Fernandez, N.-C. Nguyen, J. Peraire, Subgrid-scale modeling and implicit numerical dissipation in DG-based Large-Eddy Simulation, in: 23rd AIAA Computational Fluid Dynamics Conference, AIAA AVIATION Forum, (AIAA 2017-3951), Denver, Colorado, pp. 1–23.
[52] C. Talnikar, P.J. Blonigan, J. Bodart, Q. Wang, Optimization with LES – algorithms for dealing with sampling error of turbulence statistics, in: 53rd AIAA Aerospace Sciences Meeting, AIAA SciTech Forum, (AIAA 2015-1954), Kissimmee, Florida, pp. 1–11.
[53] T.A. Oliver, N. Malaya, R. Ulerich, R.D. Moser, Estimating uncertainties in statistics computed from direct numerical simulation, Phys. Fluids 26 (2014) 035101.
[54] C. Wieselsberger, New data on the laws of fluid resistance, Phys. Z. 22 (1921) 321–328.
[55] A. Roshko, Experiments on the flow past a circular cylinder at very high Reynolds number, J. Fluid Mech. 10 (1961) 345–356.
[56] D.J. Tritton, Experiments on the flow past a circular cylinder at low Reynolds numbers, J. Fluid Mech. 6 (1959) 547–567.