Imperial College London

Efficient Sensitivity Analysis of Chaotic Systems and Applications to Control and Data Assimilation

Karim Shawki

Department of Aeronautics, Imperial College London

September 2020

Submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

Declaration

I, Karim Shawki, hereby declare that the work presented in this thesis is a result of my research for the degree of Doctor of Philosophy at Imperial College London. References to works from other sources have been fully acknowledged.

Karim Shawki

Abstract

Sensitivity analysis is indispensable for aeronautical engineering applications that require optimisation, such as flow control and aircraft design. The adjoint method is the standard approach for sensitivity analysis, but it cannot be used for chaotic systems. This is due to the high sensitivity of the system trajectory to input perturbations; a characteristic of many turbulent systems. Although the instantaneous outputs are sensitive to input perturbations, the sensitivities of time-averaged outputs are well-defined for uniformly hyperbolic systems, but existing methods to compute them cannot be used. Recently, a set of alternative approaches based on the shadowing property of dynamical systems was proposed to compute sensitivities. These approaches are computationally expensive, however. In this thesis, the Multiple Shooting Shadowing (MSS) [1] approach is used, and the main aim is to develop computational tools to allow for the implementation of MSS to large systems.

The major contributor to the cost of MSS is the solution of a linear matrix system. The matrix has a large condition number, and this leads to very slow convergence rates for existing iterative solvers. A preconditioner was derived to suppress the condition number, thereby accelerating the convergence rate. It was demonstrated that for the chaotic 1D Kuramoto Sivashinsky equation (KSE), the rate of convergence was almost independent of the #DOF and the trajectory length. Most importantly, the developed solution method relies only on matrix-vector products.

The adjoint version of the preconditioned MSS algorithm was then coupled with a gradient descent method to compute a feedback control matrix for the KSE. The adopted formulation allowed all matrix elements to be computed simultaneously. Within a single iteration, a stabilising matrix was computed. Comparisons with standard linear quadratic theory (LQR) showed remarkable similarities (but also some differences) in the computed feedback control kernels.

A preconditioned data assimilation algorithm was then derived for state estimation purposes. The preconditioner was again shown to accelerate the rate of convergence significantly. Accurate state estimations were computed for the Lorenz system.

Acknowledgements

Firstly, I would like to thank my supervisor, Dr. George Papadakis, for his continuous support and guidance over the past four years. I have gained a lot of useful knowledge and experience while working with him. Without his help, this work would have been impossible to carry out.

I would like to extend my gratitude to Al Alfi Foundation, for providing me with a PhD scholarship. I am very grateful to Mrs. Nermeen Abou Gazia and Mrs. Neamat Salem, who always treated me like family.

A special thanks goes to my family and friends in London and Cairo, with whom I've shared unforgettable moments.

Lastly but most importantly, I would like to thank my wonderful parents for their never-ending love and support. They gave me the motivation, courage and support to start and finish my PhD. I am forever grateful and in debt for everything they have done for me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation for the Present Work

Engineers are often interested in finding the sensitivity (derivative) of a system's output to changes in its inputs. For example, in the context of fluid mechanics, and considering the flow around an airfoil, one may want to know how the lift or drag varies with small changes in the coordinates of the airfoil geometry. This field of study is called *sensitivity analysis* and has numerous useful applications. Some examples are provided below.

It is possible to compute such sensitivities using finite differences. This approach requires an integration of the system equations for each perturbed input variable, and this is straightforward and suitable when the number of input variables is small. If, however, the number of input variables is large, this approach becomes impractical because it would require a separate integration for each perturbed variable. In such cases, the adjoint method is a much more efficient approach. The sensitivities with respect to multiple input variables can be computed simultaneously with a single (forward-in-time) integration of the non-linear equations set, and a single (backward-in-time) integration of the linear set of adjoint equations. The computational costs of integrating the non-linear system and linear adjoint are similar [2, 3], making this a very attractive approach.

Engineers rely on the adjoint method for design optimisation [2, 4], robust control under design uncertainties [5, 6] and optimal control [7, 8, 9, 10, 11]. The computed sensitivities are commonly used to find, iteratively, the set of design or control variables that minimise a prescribed objective. Other applications of the adjoint method include data assimilation [12, 13, 14, 15], which is the process of fusing limited and noisy experimental data with known models that contain some uncertainty (for example, the inlet conditions and parameter values), to extract reliable estimations of the true system behaviour. Some researchers have also used the adjoint method to guide grid adaption for more efficient solutions to partial differential equations and for the error correction of system objectives [16, 17]. This allows typically expensive simulations to be run on coarser grids while using the adjoint to correct the error in the objective (due to the coarse grid discretisation). This approach may result in considerable computational cost savings when many simulations are required.

The efficiency of the adjoint method and its numerous applications have made it a very popular approach. Nevertheless, it has been shown that the adjoint method fails [18] for non-linear systems characterised as *chaotic*. In this context, failure refers to the exponential growth of the adjoint variables during the backward-in-time integration, making them unsuitable for any application. The instantaneous outputs of chaotic systems are known to be extremely sensitive to very small changes in the inputs (either initial conditions or system parameters). However, under certain conditions (to be discussed in Section 1.4), the sensitivities of time-averaged outputs of chaotic systems are well-defined. In the context of fluid mechanics, most transitional or turbulent flows are chaotic, for example, bypass transition on flat plates [10], channel flows [19], flows around cylinders [20] and airfoil vortex shedding at high Reynolds number [21].

Given the importance of the adjoint method, as well as the chaotic nature of turbulent flow systems, finding useful adjoints for chaotic systems is an active field of research. Least Squares Shadowing (LSS) [22] was recently proposed as an alternative to the standard adjoint method. It has been shown to compute accurate sensitivities of long-time averaged objectives for many chaotic systems. It has also started to be used for control [23] and error correction [24, 25]. Its computational cost currently prevents its application to large chaotic systems with many positive Lyapunov exponents (to be defined in the next section).

The main aim of this thesis is to develop efficient numerical tools to allow for the large scale implementation of shadowing for computing sensitivities of time-averaged objectives and to subsequently use the developed algorithm for optimal control and data assimilation. Some necessary background information is covered in this chapter. In Section (1.2), chaotic systems are introduced, and the standard adjoint method for dynamical systems is derived in Section (1.3). The existence of well-defined sensitivities and the failure of the adjoint method for chaotic systems are discussed in Sections (1.4) and (1.5), respectively. A literature review of novel sensitivity analysis methods for chaotic systems is provided in Section (1.6). The thesis aims and objectives are detailed in Section (1.7) and the thesis outline in Section (1.8).

1.2 Chaotic Systems

In this section, some definitions and concepts for dynamical systems are reviewed, followed by a description of chaotic systems. The exposition covers only the material necessary to understand the present and the following chapters. More detailed and rigorous presentations can be found in [26, 27].

Consider a dynamical system that has N states or degrees of freedom $\mathbf{u}(t) = \begin{bmatrix} u_1(t) & u_2(t) & \dots & u_N(t) \end{bmatrix}^T$. It is assumed that the time evolution of $\mathbf{u}(t)$ in the interval $[0, T]$ can be obtained by integrating a set of ordinary differential equations (ODE)

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}, s), \quad \mathbf{u}(0) = \mathbf{u}_0 \tag{1.1}$$

The vector $\mathbf{f}(\mathbf{u}, s)$ is non-linear, and may have been obtained by the discretisation of a set of partial differential equations that describe the evolution of the system in physical space or Fourier space, for example. The system parameter(s) are denoted by s. Starting at $t = 0$, the values of $\mathbf{u}(t)$ trace a trajectory in *phase space*, as shown for example in Figure (1.1) for a system with $N = 2$. When equation (1.1) is

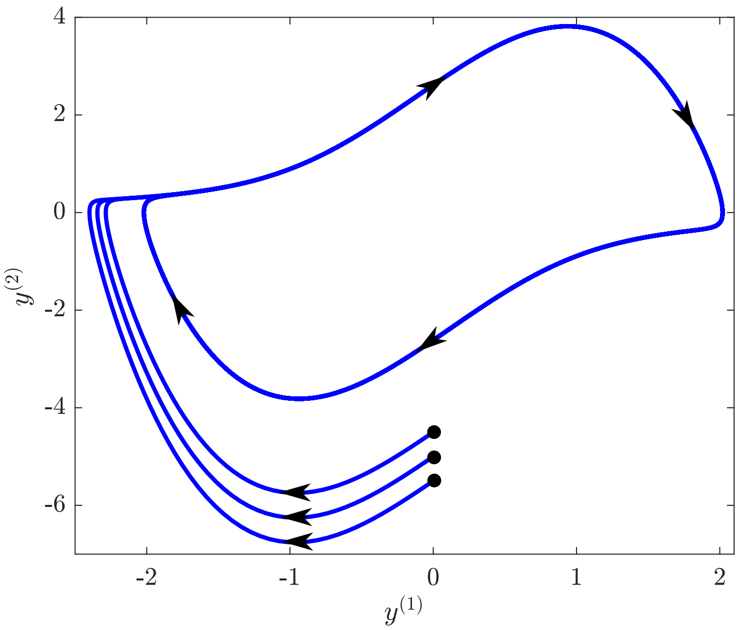


Figure 1.1: Three trajectories of the Van der Pol Oscillator ($\dot{y}^{(1)} = y^{(2)}$, $\dot{y}^{(2)} = 2(1 - y^{(1)2})y^{(2)} - y^{(1)}$) in phase space, with different initial conditions $\mathbf{u}_0$ (shown in black dots). This is an example of a stable limit cycle attractor, for which any $\mathbf{u}_0$ approaches the attractor and remains on it for $T \to \infty$.

integrated long enough, for many systems $\mathbf{u}(t)$ eventually settles on an *attractor*; a bounded region in phase space. Trajectories settle on a particular attractor if $\mathbf{u}(0)$ is located in the *basin of attraction* of that attractor. For stable linear systems, this basin encompasses all possible $\mathbf{u}(0)$, whereas non-linear systems generally have smaller basins. The time it takes for $\mathbf{u}(t)$ to reach the attractor is called the *transient* phase. The simplest attractor is a *fixed point*; in this case, the solution reaches a steady-state, and the trajectory terminates to a point in phase space (see Figure 1.2). Fixed points may be either stable or unstable to perturbations (the point in Figure 1.2 is stable). Another common attractor is the *limit cycle*; a closed orbit corresponding to a periodic trajectory $\mathbf{u}(t)$ (see Figure 1.1). Limit cycles may also be stable or unstable (nearby trajectories may spiral towards or away from them, respectively).

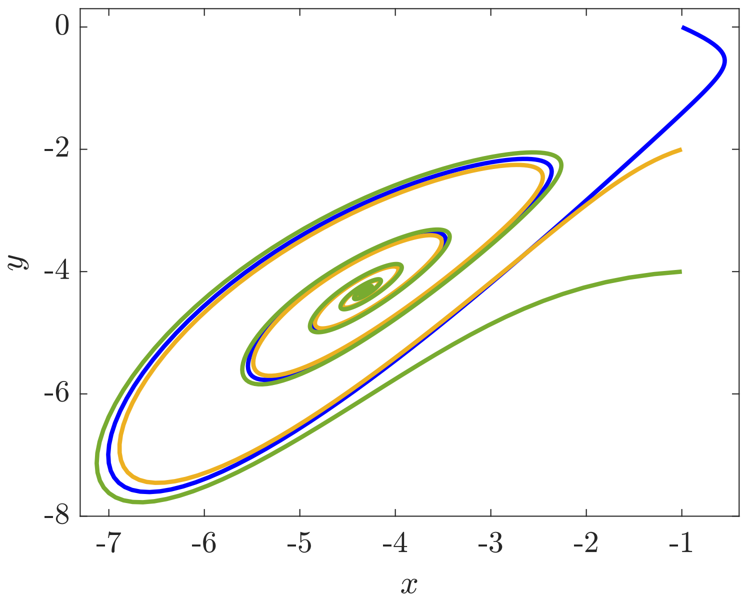


Figure 1.2: Stable fixed point attractor for the Lorenz system (with the parameter values $\sigma = 10$, $\rho = 8$ and $\beta = 8/3$).

Chaotic systems are known for their high sensitivity to small changes in $\mathbf{u}(0)$ and s. A trajectory with $\mathbf{u}(0) = \mathbf{u}_0 + \delta\mathbf{u}_0$ diverges exponentially from one with $\mathbf{u}(0) = \mathbf{u}_0$, even if $\delta\mathbf{u}_0$ is infinitesimally small. This is popularly known as the *butterfly effect* (see Figure 1.3). This trajectory divergence can be expressed mathematically as

$$\|\delta\mathbf{u}(t)\| \sim e^{\lambda_{max} t}\|\delta\mathbf{u}_0\| \tag{1.2}$$

where λ_{max} is known as the *maximal Lyapunov exponent* ($\lambda_{max} > 0$ for chaotic

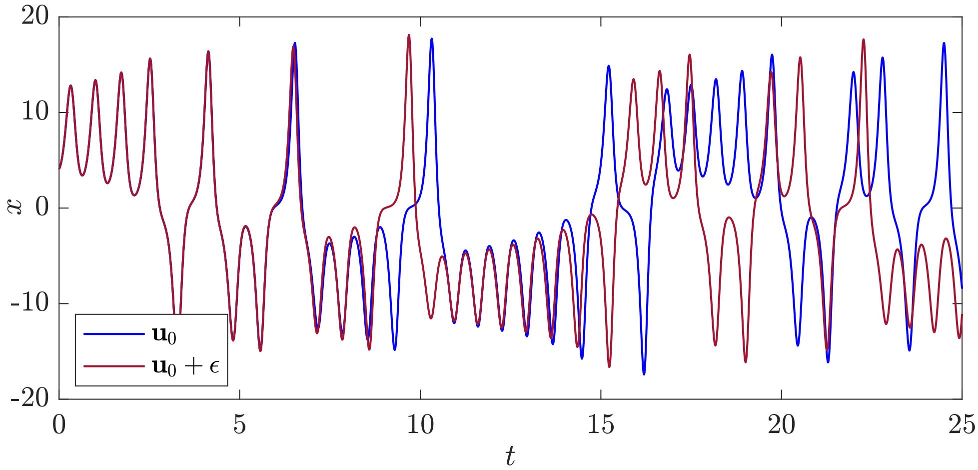


Figure 1.3: Time histories of the variable $x(t)$ of the chaotic Lorenz System ($dx/dt = \sigma(y-x)$, $dy/dt = x(\rho - z) - y$, $dz/dt = xy - \beta z$), computed with slightly different initial conditions ($\mathbf{u}_0$ and $\mathbf{u}_0 + \epsilon = \mathbf{u}_0 + 0.001$). The parameter values used are $\sigma = 10$, $\rho = 28$ and $\beta = 8/3$.

systems), and $\| * \|$ denotes a norm to quantify the distance in phase space. A similar divergence also occurs if the parameter s is perturbed. This divergence is an inherent feature of chaotic systems, i.e. trajectories always diverge regardless of the accuracy of the numerical schemes used in integrating (1.1).

The *Lyapunov time* is defined as

$$T_{lyap} = \frac{1}{\lambda_{max}} \tag{1.3}$$

and is the time taken for an initial perturbation (or error in $\mathbf{u}(0)$) to grow by a factor of e. For chaotic systems, the instantaneous dynamics are only predictable up to a time scale of $O(T_{lyap})$. It is important to note that a perturbed trajectory diverges locally in phase space, while globally it follows a different path in the same confined area (of the attractor). In other words, perturbed trajectories diverge and approach one another repeatedly. These two properties are necessary for a system to be characterised as chaotic. The type of attractor associated with chaotic systems is called a *strange attractor* (Figure 1.4).

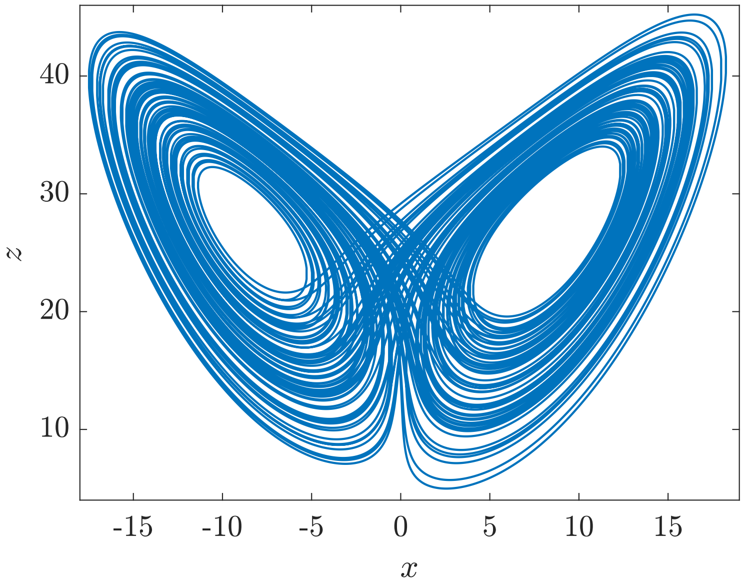


Figure 1.4: The strange attractor of the Lorenz system, computed using the standard parameters $\sigma = 10$, $\rho = 28$ and $\beta = {}^{8}\!/\!_{3}$.

Lyapunov vectors and exponents

A dynamical system with N degrees of freedom has N Lyapunov vectors $\Xi = \begin{bmatrix} \boldsymbol{\xi}_1(t) & \boldsymbol{\xi}_2(t) & \dots & \boldsymbol{\xi}_N(t) \end{bmatrix}$ (also known as covariant Lyapunov vectors), with N corresponding Lyapunov exponents $\Lambda = \begin{bmatrix} \lambda_1 & \lambda_2 & \dots & \lambda_N \end{bmatrix}$, where $\lambda_1 > \lambda_2 > \dots > \lambda_N$. Lyapunov vectors correspond to the directions of growth/decay of an initial perturbation to the system at exponential rates corresponding to the Lyapunov exponents. Suppose that the initial condition of (1.1) is perturbed by $\delta\mathbf{u}_0$. If at some time t, $\boldsymbol{\xi}_i(t)$ is parallel to $\delta\mathbf{u}_0$, the perturbation would shrink exponentially if $\lambda_i < 0$, or grow exponentially if $\lambda_i > 0$, in the direction of $\boldsymbol{\xi}_i(t)$.

Stable fixed point attractors have all $\lambda_i < 0$, and any perturbation to the system decays exponentially towards the fixed point. Limit cycle attractors have $\lambda_1 = 0$ and $\lambda_2, \lambda_3, \dots, \lambda_N < 0$. The vector corresponding to the zero (neutral) exponent is tangent to the limit cycle trajectory. Any perturbation parallel to $\boldsymbol{\xi}_1$ is tangent to the limit cycle, while perturbations parallel to other vectors decay exponentially.

Chaotic systems have at least one positive Lyapunov exponent. The number of positive exponents is denoted by N_{+LE}. Perturbations grow/shrink in directions and rates corresponding to $\boldsymbol{\xi}_i(t)$ and λ_i, respectively, however, the growth rate is eventually dominated by $\lambda_1 = \lambda_{max}$.

To quantify λ_i, consider the evolution equation for $\delta\mathbf{u}(t) = \mathbf{u}'(t, \mathbf{u}_0 + \delta\mathbf{u}_0) - \mathbf{u}(t, \mathbf{u}_0)$ (the dash $'$ refers to a trajectory with a slightly perturbed initial condition), which can be found by linearising (1.1) around $\mathbf{u}(t, \mathbf{u}_0)$,

$$\frac{d(\delta\mathbf{u})}{dt} = \mathbf{f}(\mathbf{u}') - \mathbf{f}(\mathbf{u}) = \left.\frac{\partial\mathbf{f}}{\partial\mathbf{u}}\right|_{\mathbf{u}(t)} \delta\mathbf{u}(t) + \dots \tag{1.4}$$

where $\left.{}^{\partial\mathbf{f}}/{\partial\mathbf{u}}\right|_{\mathbf{u}(t)}$ is the Jacobian matrix evaluated along the trajectory with the initial condition $\mathbf{u}_0$. The following equation,

$$\frac{d(\delta\mathbf{u})}{dt} = \left.\frac{\partial\mathbf{f}}{\partial\mathbf{u}}\right|_{\mathbf{u}(t)} \delta\mathbf{u}(t), \quad \delta\mathbf{u}(0) = \delta\mathbf{u}_0 \tag{1.5}$$

is referred to as the *homogeneous tangent* equation. This is a linear equation describing the separation of two trajectories in time with initial perturbation $\delta\mathbf{u}(0) = \delta\mathbf{u}_0$, assuming that $\delta\mathbf{u}_0$ is small and that $\delta\mathbf{u}(t)$ is within the linear range. From (1.2), λ_{max} can be written as

$$\lambda_{max} = \max_{\delta\mathbf{u}_0} \lim_{t\to\infty} \frac{1}{t} \ln \frac{\|\delta\mathbf{u}(t)\|_2}{\|\delta\mathbf{u}_0\|_2} \tag{1.6}$$

The solution to (1.5) is given analytically by,

$$\delta\mathbf{u}(t) = \phi^{t_0,t}\delta\mathbf{u}_0 \tag{1.7}$$

where $\phi^{t_0,t} = \phi^{0,t}$ is the *state transition matrix* satisfying

$$\frac{d\phi^{t_0,t}}{dt} = \left(\left.\frac{\partial\mathbf{f}}{\partial\mathbf{u}}\right|_t\right)\phi^{t_0,t}, \quad \phi^{t_0,t_0} = I \tag{1.8}$$

The matrix $\phi^{t_0,t}$ maps any initial condition $\delta\mathbf{u}_0$ at an initial time t_0 to the final time t when the product $\phi^{t_0,t}\delta\mathbf{u}_0$ is taken. Computing $\phi^{t_0,t}$ is however unnecessary, and time-steppers for (1.5) are used instead to compute the matrix-vector product $\phi^{t_0,t}\delta\mathbf{u}_0$. By substituting (1.7) into (1.6), one arrives at

$$\lambda_{max} = \lim_{t\to\infty} \frac{1}{2t} \ln\left(\hat{n}^T \phi^T \phi \hat{n}\right) \tag{1.9}$$

where $\hat{n} = {}^{\delta\mathbf{u}_0}/_{\|\delta\mathbf{u}_0\|_2}$. By rewriting ϕ in terms of its singular value decomposition (SVD), it can be shown [28] that λ_{max} is given by

$$\lambda_{max} = \lim_{t\to\infty} \frac{1}{t} \ln(\sigma_{max}(\phi)) \tag{1.10}$$

where $\sigma_{max}(\phi)$ is the largest singular value of ϕ. The remaining exponents can be computed similarly through [28]

$$\lambda_i = \lim_{t\to\infty} \frac{1}{t} \ln(\sigma_i(\phi)) \tag{1.11}$$

If the time limit ($t \to \infty$) in (1.11) is not applied, one can compute the finite-time exponents

$$\lambda_i^{'}(\mathbf{u}_0, t) = \frac{1}{t} \ln(\sigma_i(\phi)) \tag{1.12}$$

describing the local growth/shrink rates along a trajectory, which depend on t and $\mathbf{u}_0$. Lyapunov exponents can also be computed using QR decomposition, as shown in [29]. In practice, algorithms based on SVD and QR decomposition require the integration of (1.5) over short integration segments, and orthogonalisation of the initial conditions for each integration segment.

1.3 Sensitivity Analysis of Dynamical Systems

In this section, the standard sensitivity analysis approach of dynamical systems is reviewed. For simplicity, it is initially assumed that the dynamical system (1.1) has a single parameter, s. Consider a system output (or objective function) denoted by $J(\mathbf{u}(t,s),s)$. The objective function depends explicitly on $\mathbf{u}(t,s)$, while the dependence on s can be implicit or explicit (or both). The long-time averaged objective is written as

$$\overline{J^{(\infty)}} = \lim_{T\to\infty} \frac{1}{T}\int_0^T J(\mathbf{u}(t,s),s)\,dt \tag{1.13}$$

The quantity $\overline{J^{(\infty)}}$ is referred to simply as the *objective*. When $J(\mathbf{u}(t,s))$ is averaged over finite T, i.e.,

$$\overline{J^{(T)}} = \frac{1}{T}\int_0^T J(\mathbf{u}(t,s),s)\,dt \tag{1.14}$$

the notation $\overline{J^{(T)}}$ is used. To find the sensitivity $d\overline{J^{(\infty)}}/ds$, the chain rule is applied to (1.13):

$$\frac{d\overline{J^{(\infty)}}}{ds} = \lim_{T\to\infty}\frac{1}{T}\int_0^T \left(\frac{\partial J(\mathbf{u},s)}{\partial \mathbf{u}}^T \mathbf{v}(t,s) + \frac{\partial J(\mathbf{u},s)}{\partial s}\right) dt \tag{1.15}$$

where the vector $\mathbf{v}(t,s) = d\mathbf{u}/ds$ is the *state sensitivity*, $\partial J(\mathbf{u},s)/\partial\mathbf{u}$ is also a vector and $\partial J(\mathbf{u},s)/\partial s$ is a scalar. The vector $\mathbf{v}(t,s)$ is computed by linearising (1.1) around the reference trajectory for an infinitesimal perturbation δs,

$$\frac{d(\delta\mathbf{u})}{dt} = \mathbf{f}(\mathbf{u}+\delta\mathbf{u}, s+\delta s) - \mathbf{f}(\mathbf{u},s) = \left.\frac{\partial\mathbf{f}}{\partial\mathbf{u}}\right|_{\mathbf{u},s}\delta\mathbf{u}(t) + \left.\frac{\partial\mathbf{f}}{\partial s}\right|_{\mathbf{u},s}\delta s + \ldots \tag{1.16}$$

Dividing through by δs yields the inhomogeneous *tangent* equation

$$\frac{d\mathbf{v}}{dt} = \left.\frac{\partial\mathbf{f}}{\partial\mathbf{u}}\right|_{\mathbf{u},s}\mathbf{v} + \left.\frac{\partial\mathbf{f}}{\partial s}\right|_{\mathbf{u},s}, \quad \mathbf{v}(0) = 0 \tag{1.17}$$

The initial condition $\mathbf{v}(0) = 0$ indicates that $\mathbf{u}(0)$ remains the same for both trajectories. Equation (1.17) is integrated in $[0,T]$ then (1.15) is evaluated to give $d\overline{J^{(\infty)}}/ds$.

Equations (1.17) and (1.5) differ only in their initial conditions and forcing terms (equation (1.17) has the forcing term $\partial \mathbf{f}/\partial s$, i.e. it is inhomogeneous, while equation (1.5) is homogeneous).

If the sensitivity of $\overline{J^{(\infty)}}$ to multiple parameters is required, applying (1.17) would become very expensive, since it would have to be integrated once for each parameter (each integration with a different forcing term $\partial \mathbf{f}/\partial s$). To make the cost independent of the number of parameters, the adjoint approach is used. For this approach, equation (1.15) is augmented with (1.17) as constraints,

$$\frac{d\overline{J^{(\infty)}}}{ds} = \lim_{T\to\infty} \frac{1}{T} \int_0^T \left(\frac{\partial J}{\partial \mathbf{u}}^T \mathbf{v} + \frac{\partial J}{\partial s} - \boldsymbol{\lambda}^T \left(\frac{d\mathbf{v}}{dt} - \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \mathbf{v} - \frac{\partial \mathbf{f}}{\partial s} \right) \right) dt \tag{1.18}$$

where $\boldsymbol{\lambda}(t)$ is a vector of adjoint variables. Using integration by parts,

$$\int_0^T \boldsymbol{\lambda}^T \dot{\mathbf{v}}\, dt = [\boldsymbol{\lambda}^T \mathbf{v}]_0^T - \int_0^T \dot{\boldsymbol{\lambda}}^T \mathbf{v}\, dt \tag{1.19}$$

Substituting the right-hand-side of (1.19) into (1.18), and grouping the terms with $\mathbf{v}$ gives

$$\frac{d\overline{J^{(\infty)}}}{ds} = \lim_{T\to\infty} \frac{1}{T} \int_0^T \left(\frac{\partial \mathbf{f}}{\partial s}^T \boldsymbol{\lambda} + \frac{\partial J}{\partial s} + \mathbf{v}^T \left(\frac{d\boldsymbol{\lambda}}{dt} + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}^T \boldsymbol{\lambda} + \frac{\partial J}{\partial \mathbf{u}} \right) \right) dt - [\mathbf{v}^T \boldsymbol{\lambda}]_0^T \tag{1.20}$$

The term $\mathbf{v}^T(0)\boldsymbol{\lambda}(0)$ vanishes, since $\mathbf{v}(0) = 0$. If $\boldsymbol{\lambda}(t)$ satisfies

$$\frac{d\boldsymbol{\lambda}}{dt} = -\frac{\partial \mathbf{f}}{\partial \mathbf{u}}^T \boldsymbol{\lambda} - \frac{\partial J}{\partial \mathbf{u}} \tag{1.21}$$

with terminal condition $\boldsymbol{\lambda}(T) = 0$, then the sensitivity to any parameter can be computed using

$$\frac{d\overline{J^{(\infty)}}}{ds} = \lim_{T\to\infty} \frac{1}{T} \int_0^T \left(\frac{\partial \mathbf{f}}{\partial s}^T \boldsymbol{\lambda} + \frac{\partial J}{\partial s} \right) dt \tag{1.22}$$

The benefit of the adjoint approach now becomes clear; equation (1.21) must be integrated only once, therefore the cost becomes independent of the number of the parameters. Sensitivities computed using (1.15) and (1.22) are identical.

The above analysis is valid for non-chaotic systems, but fails to compute the correct sensitivities for chaotic systems. It is important first to ask whether or not sensitivities are well-defined for chaotic systems. This will be the topic of the next section. A discussion of the sensitivity analysis of chaotic systems will then follow in Section (1.5).

1.4 Are Sensitivities Well-defined for Chaotic Systems?

The instantaneous values of the objective function $J(\mathbf{u}, t)$ for the two trajectories $\mathbf{u}(s)$ and $\mathbf{u}(s + \delta s)$, evolve completely differently in time for chaotic systems. The same occurs for two trajectories with slightly different initial conditions $\mathbf{u}_0$. This is a direct result of the 'butterfly effect'. However, long time-averaged objectives $\overline{J(s)^{(\infty)}}$ of chaotic systems are often well-defined and vary smoothly with s for almost all $\mathbf{u}_0$. If the system (1.1) is *ergodic*, then $\overline{J(s)^{(\infty)}}$ is independent of the initial condition $\mathbf{u}_0$. Ergodicity is therefore a necessary condition for $d\overline{J^{(\infty)}}/ds$ to be well-defined.

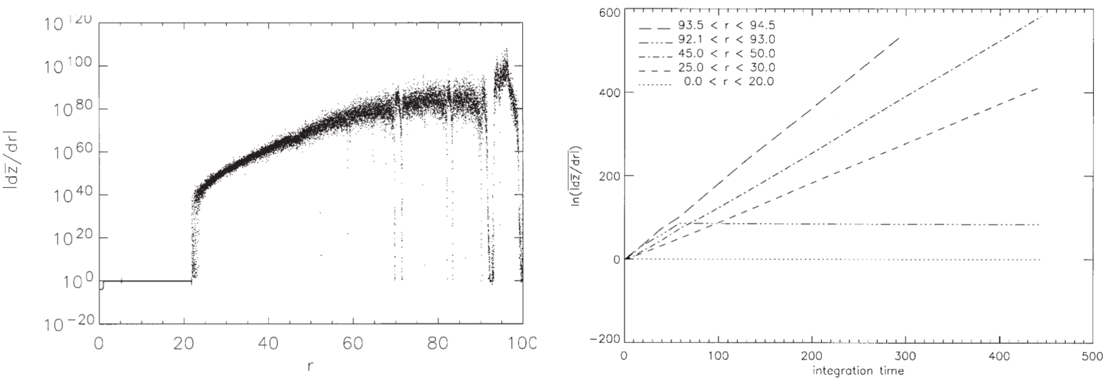
The quantity $\overline{J(s)^{(\infty)}}$ is known to be differentiable for uniformly *hyperbolic systems* [30]. For such systems, $\overline{J(s)^{(\infty)}}$ varies smoothly with change in s. Uniformly hyperbolic systems are characterised by having a tangent space that can be decomposed into stable and unstable manifolds at any given point on the attractor [31]. This means that the dynamics at any point on the attractor is either exponentially contracting or expanding. The manifolds always intersect, i.e. the angle between them is always non-zero. Non-hyperbolic systems (such as the Rössler attractor) have non-smoothly varying $\overline{J(s)^{(\infty)}}$ with s, meaning that sensitivities are not well-defined (as shown in [32]). In conclusion, for the sensitivity of $\overline{J(s)^{(\infty)}}$ to be well-defined, the system needs to be ergodic and uniformly hyperbolic. An important property of hyperbolic systems called the 'shadowing' property has been used to derive a family of methods to compute the sensitivity. More details will be given in Chapter (2).

Systems with hyperbolic attractors are very uncommon. However, Ruelle's linear response theorem [33] states that some properties of hyperbolic systems, such as the differentiability property, may apply to non-hyperbolic attractors as well, such as *quasi-hyperbolic* attractors. Quasi-hyperbolic attractors can have some *homoclinic tangencies*, meaning that the Lyapunov vectors can become tangent at some points along the attractor, which means that the system is not strictly hyperbolic. A common example of a quasi-hyperbolic attractor is the Lorenz system (for a range of parameter values).

Larger turbulent systems may behave more like hyperbolic systems according to the chaotic hypothesis of Gallavotti and Cohen [34]. This means that well-defined sensitivities may exist for many large turbulent systems. Only a few studies have been carried out to date to confirm this, however. Recently, Ni [35] showed that sensitivities exist and may be computed using the shadowing method (see Chapter 2) for a 3D flow around a rotating cylinder at $Re = 525$.

1.5 Failure of the Adjoint Method for Chaotic Systems

The failure of the standard adjoint method (Section 1.3) is discussed in this section. Lea et al. [36] demonstrated this failure numerically. They used the adjoint equation to compute $d\overline{z^{(T)}}/d\rho$ for the Lorenz system. They computed $d\overline{z^{(T)}}/d\rho$ for a range of ρ values, as shown in Figure (1.5a). The reference values (computed by finite differences) are $\overline{dz^{(T)}}/d\rho \approx 1$. For $\rho < 24$, the correct sensitivities were computed, while for $\rho > 24$, $\overline{dz^{(T)}}/d\rho$ was many orders of magnitude larger than expected. At $\rho \approx 24$, the Lorenz attractor transitions from having two stable fixed points ($\lambda_{max} < 0$) to a strange attractor ($\lambda_{max} > 0$), explaining the sudden surge in $\overline{dz^{(T)}}/d\rho$ at that parameter value. In Panel (b), $\ln\left(|\overline{dz^{(T)}}/d\rho|\right)$ is shown against time for different ρ value ranges. The gradients of the lines agreed with the corresponding λ_{max} values, demonstrating that $|\overline{dz^{(T)}}/d\rho| \propto \exp(\lambda_{max} t)$. Similar behaviour can be expected for other chaotic systems.



(a) Sensitivity $|\overline{dz^{(T)}}/d\rho|$ vs. parameter ρ (b) Logarithm of $|\overline{dz^{(T)}}/d\rho|$ vs. time T

Figure 1.5: Failure of the adjoint equation (1.21) to compute $\overline{dz^{(T)}}/d\rho$ for the Lorenz system. Note that $r = \rho$ in the notation of [36]. Reprinted from [36], with permission from Taylor & Francis.

The failure of the adjoint approach for chaotic systems can now be explained. Let's assume that $\overline{dJ^{(\infty)}}/ds$ is well-defined, i.e. that the system is ergodic and hyperbolic. Computing $\overline{dJ^{(\infty)}}/ds$ for general dynamical systems was discussed in Section (1.3). By linearising (1.1) due to perturbations in s, the sensitivity $\overline{dJ^{(\infty)}}/ds$ may be computed for one parameter (by integrating the tangent equation (1.17) and evaluating (1.15)) or simultaneously for many parameters (by integrating the adjoint equation (1.21) once and evaluating (1.22) for each parameter).

As previously mentioned, the above approaches are invalid for chaotic systems. Equation (1.15) assumes that the two limiting operations ($\delta s \to 0$ and $T \to \infty$)

commute, but this is not valid for chaotic systems [22]. The following inequality applies:

$$\lim_{\delta s\to 0}\lim_{T\to\infty}\frac{1}{T}\int_0^T\frac{J(s+\delta s)-J(s)}{\delta s}\,dt\neq\lim_{T\to\infty}\frac{1}{T}\int_0^T\lim_{\delta s\to 0}\frac{J(s+\delta s)-J(s)}{\delta s}\,dt \quad (1.23)$$

where the right-hand-sides of (1.23) and (1.15) are identical. When T is sufficiently large and $N_{+LE}\geq 1$, the tangent (1.17) variables grow exponentially forward in time, i.e.,

$$\mathbf{v}(t)-\mathbf{v}(0)\sim e^{\lambda_{max}t} \quad (1.24)$$

while the adjoint (1.21) variables grow exponentially backward in time, i.e.,

$$\boldsymbol{\lambda}(t)-\boldsymbol{\lambda}(T)\sim e^{\lambda_{max}(T-t)} \quad (1.25)$$

As a result, the sensitivities obtained by either evaluating (1.15) or (1.22), also grow exponentially with T and quickly become meaningless.

1.6 Previous Research on Chaotic Sensitivity Analysis

This section reviews previous attempts to address the failure of standard sensitivity analysis for chaotic systems. Unsurprisingly, most attempts have been made by the 'climate' and 'turbulence' communities, for which information on sensitivities is often required. The literature review is split into two parts. The first part reviews the Ensemble Adjoint method, while the second part reviews methods designed to replace the ill-conditioned adjoint (1.21) with well-conditioned alternatives.

1.6.1 The Ensemble Adjoint method

The backward-in-time exponential growth of $\boldsymbol{\lambda}(t)$ was discussed in Section (1.5). The adjoint variables $\boldsymbol{\lambda}(t)$ grow backward in time at an exponential rate determined by λ_{max} (starting from some time $t = \tau$). Lea et al. [36] suggested splitting the trajectory into K segments of length ΔT each. If ΔT is small enough, $\boldsymbol{\lambda}(t)$ would be bounded. Each segment is treated separately as a trajectory with an initial condition and a corresponding adjoint equation (with a zero terminal condition). The sensitivity $d\overline{J^{(\Delta T)}}_i/ds$ for each segment i is computed and averaged over all segments, to provide an estimate of the long-time averaged sensitivity $d\overline{J^{(\infty)}}/ds$. This approach is known as the Ensemble Adjoint method. It was first applied to the Lorenz system in [36] and to an ocean circulation model in [18]. Eyink et al. [37] derived an equivalent method which uses Ruelle's linear response formula. Although ensemble adjoint methods have been shown in some cases to produce results with 'reasonable' accuracy, the choice of ΔT is tricky, and cannot be easily estimated in advance. If ΔT is too 'small', sensitivities may become biased (since the averaging time is too short). If ΔT is too 'large', $\boldsymbol{\lambda}(t)$ may grow boundlessly. Furthermore, the convergence of sensitivities has been shown in [38] to be extremely slow (K must be intractably large for convergence). These reasons make the Ensemble Adjoint method difficult to use in practice.

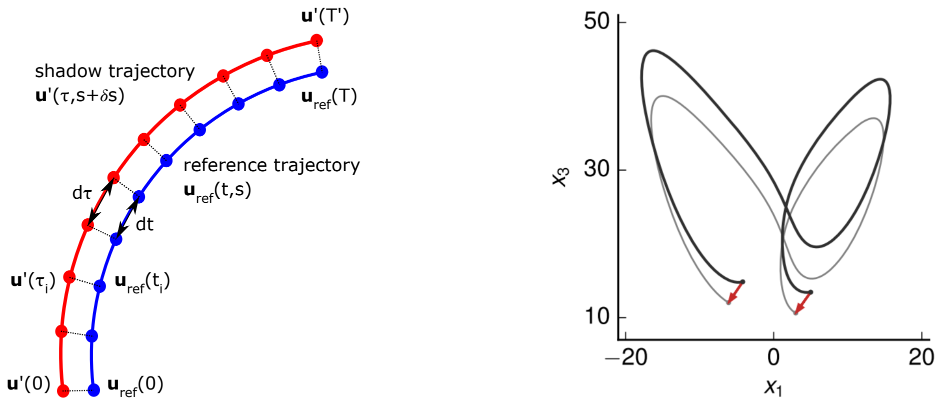
1.6.2 Computing bounded adjoint variables

Several methods proposed to replace or reformulate the ill-conditioned tangent (1.17) and adjoint (1.21) equations are discussed below. The general idea is to formulate well-conditioned tangents/adjoints that remain bounded in time, such that $d\overline{J^{(T)}}/ds$ can be computed accurately.

Thuburn [38] expressed $\overline{J(s)^{(\infty)}}$ in terms of the probability density function (PDF), which is governed by the Fokker-Plank equation. An adjoint equation was derived, allowing simultaneous computation of sensitivities. However, the requirement to discretise in phase space and the potential inaccuracy of the sensitivities (due to the addition of an artificial diffusion term to smooth the solution), make this method difficult to apply to large systems.

Lasagna [39] replaced the non-linear equation set (1.1) with a time-periodic boundary value problem (BVP), which computes unstable periodic orbits (UPO) with different periods. As a result, the corresponding tangent variables (obtained by linearising around the UPO) and the adjoint variables, are also periodic and bounded in time. Sensitivities were computed accurately for two chaotic systems and were found to be mostly independent of the length of the UPO used. The application of this method to turbulent systems is challenging, however. This is because computing UPOs becomes progressively more challenging with increasing Reynolds number [40].

Another set of methods uses the *shadowing* property to compute sensitivities. In this section, the main idea is briefly described. The method will be formally presented and discussed in the next chapter. The idea is to find a perturbed shadow trajectory $\mathbf{u}'(s + \delta s)$ that remains close in phase space to a reference trajectory $\mathbf{u}_{ref}(s)$, both satisfying the non-linear equation set (1.1). The two trajectories have different initial conditions, but for an ergodic system, this does not affect the computed objective (or its sensitivity). Since $\mathbf{u}'$ (the shadow trajectory) does not deviate from $\mathbf{u}_{ref}$, it can be used to compute useful sensitivities. The existence of $\mathbf{u}'$ is guaranteed for hyperbolic systems according to the shadowing Lemma [41] (to be discussed in Section 2.1) but has been shown to exist in practice for quasi-hyperbolic systems as well [22]. One method [22] computes $\mathbf{u}'(\tau, s + \delta s)$ by minimising the distance $\|\mathbf{u}'(\tau) - \mathbf{u}_{ref}(t)\|$ in time. This is the Least Squares Shadowing (LSS) [22] approach. An adjoint has been derived using LSS. Another approach, called Periodic Shadowing [42], derives a time-periodic tangent by enforcing an equal distance between $\mathbf{u}'$ and $\mathbf{u}_{ref}$ at the endpoints of the attractor (i.e. $\mathbf{u}'(0) - \mathbf{u}_{ref}(0) = \mathbf{u}'(T) - \mathbf{u}_{ref}(T)$). The concepts of LSS and Periodic Shadowing are shown in Figure (1.6).



(a) Least Squares Shadowing [22]

(b) Periodic Shadowing [42]

Figure 1.6: Two different methods to compute the shadow trajectory $\mathbf{u}'(t)$. LSS (Panel a) minimises the distance between $\mathbf{u}'(t)$ and $\mathbf{u}_{ref}(t)$ at all time instants. Periodic Shadowing (Panel b) enforces $\mathbf{u}'(0) - \mathbf{u}_{ref}(0) = \mathbf{u}'(T) - \mathbf{u}_{ref}(T)$ (as shown by the red arrows). Panel (b) is reprinted from [42], with permission from the Journal of Computational Physics.

Shadowing is arguably the most promising approach for computing the sensitivities of large chaotic systems. It has already been shown to compute accurate sensitivities for several turbulent systems (such as vortex shedding behind an airfoil [21] and 3D flow around a cylinder [20]). This method is however expensive and requires significant cost reductions in order to make it applicable to large turbulent flow systems with large N_{+LE}. Both LSS and Periodic Shadowing are BVPs, requiring multiple shooting type methods for large systems. Such methods for Periodic Shadowing can be found in [42], and for LSS in [1, 43]. Multiple Shooting Shadowing (MSS) [1] is memory efficient but requires the solution to a linear matrix system that has a large condition number, and therefore suffers from slow convergence when iterative methods are applied. Finding a way to accelerate the convergence of MSS and make it independent of the trajectory length and the number of states N, could enable the application of adjoint sensitivity analysis to very large turbulent systems.

1.7 Thesis Aims and Objectives

The overall aim of this thesis is to further develop shadowing, to accelerate the computation of sensitivities $d\overline{J^{(\infty)}}/ds$ for large chaotic systems and to use the developed algorithm for optimal control and data assimilation. The main objectives are:

1. Design a preconditioner to accelerate the convergence rate of the Multiple Shooting Shadowing method.
2. Design an optimal feedback controller based on preconditioned MSS and compare the performance with linear quadratic regulator (LQR) control theory.
3. Apply the developed preconditioner to data assimilation for chaotic systems.

1.8 Thesis Outline

The rest of this thesis is structured as follows: Chapter (2) starts with a discussion and derivation of LSS, followed by a presentation of MSS. The main issues impeding the use of LSS and MSS for large systems are pointed out. Preconditioning of MSS is examined in Chapter (3) and a detailed analysis of the convergence and cost savings is presented for the Lorenz system and the Kuramoto Sivashinsky equation (KSE). In Chapter (4), an optimal control algorithm based on the improved preconditioned MSS is proposed. The algorithm is applied to compute a feedback controller for the KSE, and the feedback kernel and performance is compared to that computed with the standard linear quadratic regulator (LQR). In Chapter (5), a data assimilation algorithm is derived and preconditioned similarly as MSS. The algorithm is used for state reconstruction of the Lorenz system, and its performance and accuracy are assessed. A summary of the main achievements and suggestions for future work are presented in Chapter (6).

Chapter 2

Shadowing of Dynamical Systems

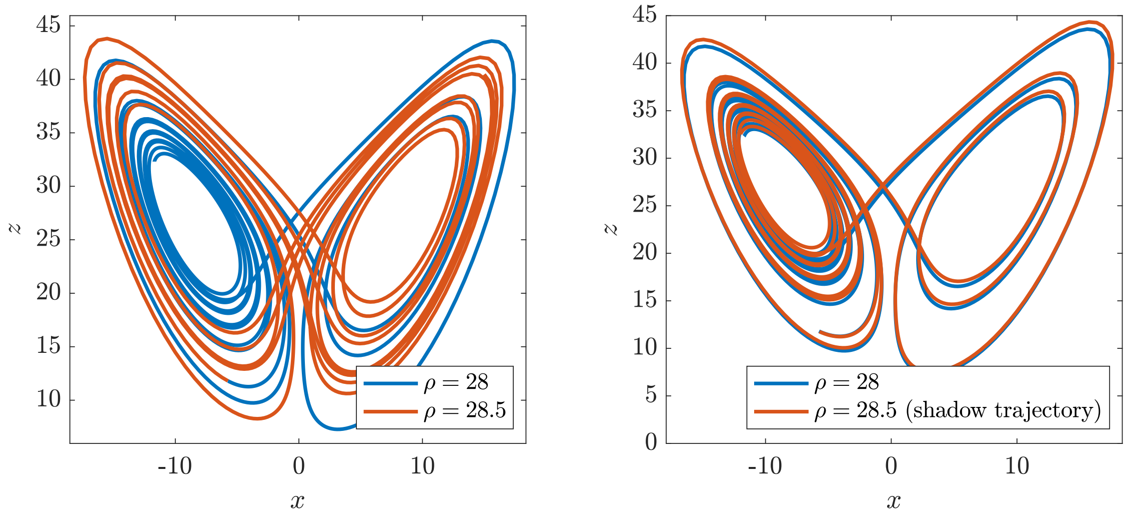
2.1 The Shadowing Lemma

The exponential divergence of two trajectories with slightly different parameter values s, as shown in Figure (2.1a), makes the standard sensitivity analysis approach (Section 1.3) inapplicable. The shadowing lemma (described below) makes it possible to find a perturbed *shadow* trajectory $\mathbf{u}'(\tau, s + \delta s)$ (see Figure 2.1b) satisfying equation (1.1), which always stays close in phase space, i.e. shadows the reference trajectory $\mathbf{u}_{ref}(t, s)$.

The shadowing lemma [41] guarantees the existence of the shadow trajectory $\mathbf{u}'$ for hyperbolic systems. To understand the concept of shadowing, consider first the discrete form solution of the non-linear ODE set, which can be written as

$$\mathbf{u}_{i+1} = g(\mathbf{u}_i) \tag{2.1}$$

where i denotes the time-step. For chaotic systems, errors for example in the time-stepping of (2.1) due to finite machine precision or round-off errors accumulate at each time-step, and this leads to the divergence of the exact and com-



(a) Divergence of 2 trajectories

(b) Shadow trajectory

Figure 2.1: A reference trajectory evaluated for the Lorenz system at $s = \rho = 28$ (in blue), and perturbed trajectories evaluated at $s = \rho = 28.5$ (in red). The perturbed trajectory in Panel (a) diverges from the reference trajectory (the two trajectories have the same initial condition). The shadow trajectory (with a slightly different initial condition) in Panel (b) shadows the reference trajectory and is computed using the method described in this chapter.

puted trajectories that are initially close in phase space. The approximate trajectory $\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_M$ obtained by time-stepping (2.1) satisfies $|\mathbf{u}_{i+1} - g(\mathbf{u}_i)| < \delta$ for $i = 0, 1, \ldots, M-1$ and is called a δ-pseudotrajectory of the system. The lemma states that there exists a true trajectory satisfying the equation exactly, i.e. $\mathbf{u}'_{i+1} = g(\mathbf{u}'_i)$ for $i = 0, 1, \ldots, M-1$ which ϵ-shadows the δ-pseudotrajectory. The distance between the trajectories is $|\mathbf{u}'_i - \mathbf{u}_i| < \epsilon$ for $i = 0, 1, \ldots, M$.

While the shadowing lemma guarantees the existence of $\mathbf{u}'$, it does not specify how $\mathbf{u}'$ can be computed. In [31], the Lagrange multiplier method was used to compute $\mathbf{u}'_i$ by minimising the distance $|\mathbf{u}'_i - \mathbf{u}_i|$ at all time-steps with the constraint $\mathbf{u}'_{i+1} - g(\mathbf{u}'_i) = 0$. It must be mentioned that the real and the approximate trajectories do not have the same initial conditions, neither the time step sizes between i and $i+1$ are equal for the two trajectories.

The same idea can be employed for sensitivity analysis, i.e. the shadowing lemma guarantees that a perturbed trajectory $\mathbf{u}'(\tau, s + \delta s)$ exists that shadows $\mathbf{u}_{ref}(t, s)$. This chapter reviews a recently proposed method based on this idea, from which a well-conditioned tangent and adjoint can be derived, allowing simultaneous computation of $d\overline{J^{(\infty)}}/ds$ for many parameters.

2.2 Least Squares Shadowing

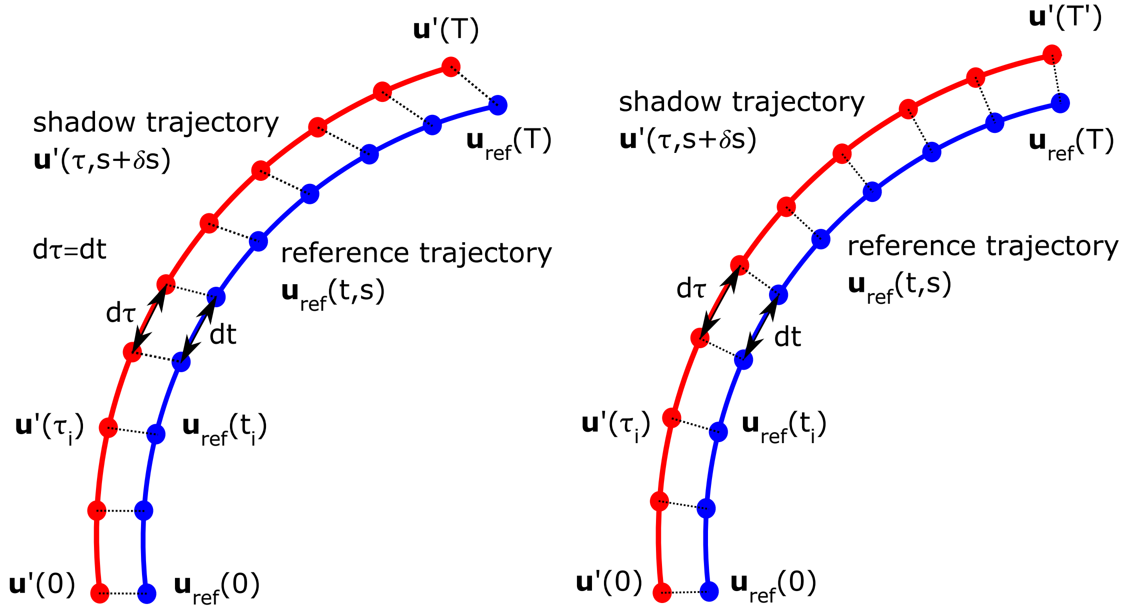
Least Squares Shadowing (LSS) was proposed by Wang et al. [22] to compute $d\overline{J^{(\infty)}}/ds$ by finding the shadow trajectory $\mathbf{u}'$. LSS makes two assumptions about the non-linear system. The first is hyperbolicity, which guarantees the existence of $\mathbf{u}'$ as discussed above. In practice, however, this requirement can be relaxed to include quasi-hyperbolic attractors as well (such as the Lorenz system), as shown in [22]. The second assumption is ergodicity. LSS relaxes the initial condition to find a new initial condition $\mathbf{u}'(0)$ such that $\mathbf{u}'$ always remains close to $\mathbf{u}_{ref}$. If the system is ergodic, this change of initial condition will not affect $\overline{J(s)^{(\infty)}}$ and its sensitivity.

Non-linear LSS [22] formulates and solves the following least squares problem:

$$\begin{aligned}
&\underset{\mathbf{u}',\tau}{\text{Minimise}} \quad \frac{1}{T}\int_0^T \left(\|\mathbf{u}'(\tau(t), s+\delta s) - \mathbf{u}_{ref}(t,s)\|_2^2 + \alpha^2 \left(\frac{d\tau}{dt} - 1\right)^2 \right) dt \\
&\text{subject to} \quad \frac{d\mathbf{u}'}{d\tau} = \mathbf{f}(\mathbf{u}', s+\delta s)
\end{aligned} \tag{2.2}$$

where δs is finite. The first term in the integral represents the distance between $\mathbf{u}'(\tau, s + \delta s)$ and $\mathbf{u}_{ref}(t, s)$, and the latter is computed from (1.1) beforehand. The second term is a 'time-stretching factor', which is required to keep the deviation $(\mathbf{u}'(\tau) - \mathbf{u}_{ref}(t))$ perpendicular to the reference trajectory (see Figure (2.2) for an illustration). The constant α is chosen such that both quantities have similar magnitude, and the optimum value is case dependent.

By taking $\delta s \to 0$ and linearising the constraint in (2.2) about $\mathbf{u}_{ref}$, the optimisation



(a) No time transformation ($^{d\tau}/_{dt} = 1$) (b) Time transformation ($^{d\tau}/_{dt} = 1 + \eta(t)$)

Figure 2.2: LSS varies the time steps $d\tau$ (Panel b), such that $(\mathbf{u}'(\tau) - \mathbf{u}_{ref}(t))$ remains perpendicular to $\mathbf{u}_{ref}$. The variable $\eta(t)$ is called the 'time dilation' term.

problem (2.2) can be written as a linear problem

$$\begin{aligned} &\underset{\mathbf{v},\eta}{\text{Minimise}} \quad \frac{1}{T}\int_0^T \|\mathbf{v}\|_2^2 + \alpha^2\eta^2\, dt \\ &\text{subject to} \quad \frac{d\mathbf{v}}{dt} = \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\mathbf{v} + \frac{\partial \mathbf{f}}{\partial s} + \eta \mathbf{f}(\mathbf{u}_{ref}, s) \end{aligned} \tag{2.3}$$

where $\mathbf{v}(t) = {}^{d(\mathbf{u}' - \mathbf{u}_{ref})}/_{ds}$ is the 'shadowing direction' and $\eta(t) = {}^{d\,(^{d\tau}/_{dt} - 1)}/_{ds}$ is the 'time dilation term'. By using calculus of variations, the solution to the minimisation problem (2.3) must satisfy

$$\frac{d\mathbf{v}}{dt} - \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\mathbf{v} - \frac{\partial \mathbf{f}}{\partial s} - \eta \mathbf{f} = 0 \tag{2.4a}$$

$$\frac{d\mathbf{w}}{dt} + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}^T \mathbf{w} - \mathbf{v} = 0 \tag{2.4b}$$

$$\mathbf{w}(0) = \mathbf{w}(T) = 0 \tag{2.4c}$$

$$\alpha^2\eta - \mathbf{w}^T\mathbf{f} = 0 \tag{2.4d}$$

where $\mathbf{w}(t)$ is a vector of adjoint variables. The system (2.4) is a two-point BVP.

Notice that the adjoint boundary conditions (2.4c) are now applied at both ends of the time interval. This is because now both $\mathbf{v}(0)$ and $\mathbf{v}(T)$ are unknown. The sensitivity is given by evaluating an integral similar to (1.15)

$$\frac{d\overline{J^{(T)}}}{ds} = \frac{1}{T}\int_0^T \left(\frac{\partial J}{\partial \mathbf{u}}^T \mathbf{v}(t) + \frac{\partial J}{\partial s} + \eta(t)(J(t) - \overline{J^{(T)}}) \right) dt \tag{2.5}$$

where the effect of the time dilation term $\eta(t)$ must be taken into account. In most LSS literature [22, 44, 45, 46, 21], the system (2.4) is discretised using finite differences, leading to a Karush-Kuhn-Tucker (KKT) saddle point system

$$\begin{bmatrix} I & & G^T \\ & \alpha^2 I & F^T \\ G & F & \end{bmatrix} \begin{bmatrix} \underline{\mathbf{v}} \\ \underline{\boldsymbol{\eta}} \\ \underline{\mathbf{w}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \underline{\mathbf{b}} \end{bmatrix} \tag{2.6}$$

where the vectors $\underline{\mathbf{v}} = \begin{bmatrix} \mathbf{v}_0^T & \mathbf{v}_1^T & \dots & \mathbf{v}_M^T \end{bmatrix}^T$, $\underline{\boldsymbol{\eta}} = \begin{bmatrix} \eta_0 & \eta_1 & \dots & \eta_M \end{bmatrix}^T$ and $\underline{\mathbf{w}} = \begin{bmatrix} \mathbf{w}_0^T & \mathbf{w}_1^T & \dots & \mathbf{w}_M^T \end{bmatrix}^T$ represent the discrete forms of $\mathbf{v}(t)$, $\boldsymbol{\eta}(t)$ and $\mathbf{w}(t)$, at equally spaced time steps $m = 0, 1, \dots, M$. The matrix G is upper block bi-diagonal and F is block diagonal. The Schur complement system of (2.6),

$$\left(GG^T + \frac{1}{\alpha^2} FF^T \right) \underline{\mathbf{w}} = \underline{\mathbf{b}} \tag{2.7}$$

is usually solved instead, because the system matrix is symmetric positive definite. An adjoint version of (2.6) with the same matrix but different right-hand-side vector was derived in [22] to compute sensitivities to many parameters simultaneously.

Solving (2.7) is very challenging with increasing N or T. The Schur complement matrix is size $MN \times MN$. For large 3D turbulent systems, values of $MN \gg 10^{10}$ are not uncommon. This means that reduced-order models (ROM) or matrix-free methods are required to keep storage requirements acceptable. To keep computational times low, efficient iterative methods to solve (2.7) must be used. The next section reviews previous attempts to compute sensitivities using LSS and to reduce its computational costs.

2.2.1 Previous applications of LSS

Previous research on LSS has focused on assessing its ability to compute sensitivities in rather small (chaotic) turbulent systems, as well as to find efficient ways to solve the LSS system (2.7). The main findings are summarised below.

The first CFD application of LSS was in homogeneous isotropic turbulence, described by the Navier-Stokes equations in wave-number space [44]. The domain considered was a cube with $Re_\lambda = 33.06$ (based on a Taylor micro-scale). The sensitivity of the time-averaged cumulative energy spectrum to a given parameter was found to agree well with finite-difference data.

In [21], LSS was applied to a flow around an airfoil experiencing chaotic vortex shedding from the suction side. The sensitivity of the time-averaged drag coefficient to the Mach number was computed. The sensitivities for different Mach numbers were found to match to within 1% compared to slope data obtained from a curve fit.

Sensitivities of the time-averaged energy of the Kuramoto Sivashinsky equation (a 4^{th} order chaotic PDE) to a system coefficient, were computed in [45]. LSS sensitivities agreed reasonably well with the reference data in the 'light turbulence regime', though a reproducible bias of 8% was reported in this regime. The origin of this bias and an approach to eliminate it will be discussed in Chapter (3). Sensitivities computed in the 'convection dominated regime' failed to match with the reference data.

Some different iterative methods were used to solve the aforementioned linear LSS

Schur complement system (2.7). Generalised Minimal Residual (GMRES) was used to solve (2.7) for the airfoil case [21]. For this small, turbulent problem (2,218 nodes and $N = 11,090$), convergence and storage requirements were enormous. For the chaotic simulations, up to 24 GB of storage data was required, and for some simulations, more than 2×10^5 iterations were needed to reduce the residual to machine zero. Since each GMRES iteration requires a product of the Schur complement with a vector, the number of floating-point operations required make this approach far too expensive.

The solution to (2.7) for the Lorenz system using multigrid-in-time was explored in [46]. This approach involved restricting (2.7) to coarse grids in time, then reducing the residual with some relaxation iterations, followed by prolongation of the solution to finer grids. It was found that a typical 'v' cycle (with injection for restriction, linear interpolation for prolongation and Gauss-Seidel for smoothing) suffered from slow convergence (10^4 cycles). The use of higher-order averaging for prolongation and restriction, as well as the use of Krylov subspace methods for smoothing, accelerated the convergence. The use of cyclic reduction required a single cycle for convergence, but the large floating point operation count would make it difficult to scale to larger systems.

The use of LSS for the sensitivity analysis of large systems remains unlikely due to the high computational costs. The next two sections describe two recently proposed variants to reduce the computational costs.

2.3 Non-Intrusive Least Squares Shadowing

Non-intrusive Least Squares Shadowing (NILSS), developed by Ni and Wang [43], reduces the cost of computing the shadowing direction $\mathbf{v}(t)$. It replaces (2.3) with a similar least squares problem having significantly fewer unknowns. The authors consider the following decomposition of the shadowing direction $\mathbf{v}(t)$:

$$\mathbf{v}(t) = \mathbf{v}^*(t) + W(t)\alpha \tag{2.8}$$

where $\mathbf{v}^*(t)$ satisfies the inhomogeneous tangent equation (1.17) and $W(t)$ is a $N \times p$ matrix. The p columns comprise of solutions $\mathbf{w}(t)$ to the homogeneous tangent (1.5) with random initial conditions, and α is an unknown weighting vector.

Integrating the tangent equation (1.17) gives the time evolution of $\mathbf{v}^* = {}^{d\mathbf{u}}/_{ds}$, i.e. the effect of perturbing (1.1) by δs under fixed $\mathbf{u}(0)$. This solution grows exponentially at a rate of λ_{max}. The homogeneous tangents $\mathbf{w}(t) = {}^{d\mathbf{u}}/_{d\mathbf{u}_0}$ described by equation (1.5), quantify the time evolution of an initial perturbation to $\mathbf{u}(0)$ of (1.1) (under fixed s) and also grow exponentially. The idea is to find the combination $W(t)\alpha$ that cancels out the exponential growth of $\mathbf{v}^*(t)$, resulting in a bounded and useful shadowing direction $\mathbf{v}(t)$. The following least squares problem finds the vector α that cancels out the fastest growing p modes:

$$\underset{\alpha}{\text{Minimise}} \quad \frac{1}{2}\int_0^T \left(\mathbf{v}^{*\perp}(t) + W^{\perp}(t)\alpha\right)^T \left(\mathbf{v}^{*\perp}(t) + W^{\perp}(t)\alpha\right) dt \tag{2.9}$$

where $\perp$ is an orthogonal projector used to extract the vector component normal to the reference trajectory. It is necessary that $p \geq N_{+LE}$, such that all unstable modes are cancelled out and a bounded shadowing direction $\mathbf{v}(t)$ is found. The above problem requires partitioning the trajectory into P segments (with segment length $O({}^1/_{\lambda_{max}})$). The solution to (2.9) requires the inversion of a Schur complement system with size $Pp \times Pp$. This matrix is many orders of magnitude smaller than (2.7) for large systems, because typically $P \ll M$ and $p \ll N$. On the other hand, the major contributor to the cost of NILSS is the computation of $\mathbf{v}^*(t)$ and especially $W(t)$ (i.e. 1 inhomogeneous tangent (1.17) and p homogeneous tangent (1.5) equation integrations are needed, respectively).

The cost of NILSS scales with N_{+LE}. A value $N_{+LE} = 17$ was found for a $Re = 525$ flow around a 3D cylinder [20], while $150 < N_{+LE} < 160$ was computed for a channel flow with $Re = 3000$ and $Re_\tau = 140$ (friction Reynolds number) [19]. Choosing p is challenging, since one cannot easily guess N_{+LE} beforehand for a given system. It was suggested in [20] to use trial and error, i.e. to compute an initial number of homogeneous tangents, and to augment $W^\perp$ progressively with more columns until all unstable modes have been included.

2.4 Multiple Shooting Shadowing

In another attempt to reduce the cost of LSS, Blonigan and Wang [1] proposed to minimise $\mathbf{v}(t)$ at $P+1$ discreet points in time, which define P segments, as shown in Figure (2.3). The segment size ΔT is constant and is $O(^1/_{\lambda_{max}})$. This method, known as Multiple Shooting Shadowing (MSS), reduces the number of unknowns to $P \times N \ll M \times N$. As will be shown, this method requires only matrix-vector products to solve the Schur complement matrix system, i.e. no matrix storage is required.

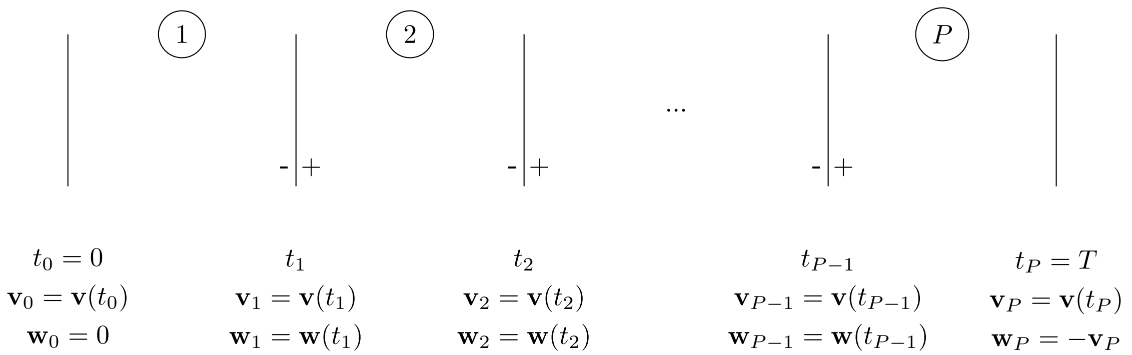


Figure 2.3: A sketch illustrating the time segmenting approach of MSS. The LSS tangent (2.4a) and adjoint (2.4b), are propagated forward and backward (respectfully) in time in the P segments, such that the continuity across segments $\mathbf{v}(t_i^+) = \mathbf{v}(t_i^-)$ is satisfied for $i = 1, 2, ..., P-1$.

2.4.1 Tangent MSS

The MSS minimisation problem can be formulated as

$$\underset{\mathbf{v}(t_i^+)}{\text{Minimise}} \quad \frac{\Delta T}{P}\sum_{i=0}^{P}\|\mathbf{v}(t_i^+)\|_2^2 \tag{2.10a}$$

$$\text{subject to} \quad \mathbf{v}(t_i^+) = \mathbf{v}(t_i^-) \qquad (i = 1, 2, ..., P-1) \tag{2.10b}$$

$$\frac{d\mathbf{v}}{dt} - \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\mathbf{v} - \frac{\partial \mathbf{f}}{\partial s} - \eta\mathbf{f} = 0 \quad t_i < t < t_{i+1} \; (i = 0, 1, ..., P-1) \tag{2.10c}$$

$$\mathbf{f}(\mathbf{u}(t), s)^T\mathbf{v}(t) = 0 \qquad t_i < t < t_{i+1} \; (i = 0, 1, ..., P-1) \tag{2.10d}$$

The above form is equivalent to the LSS minimisation problem (2.3) if $P = \infty$ and $\alpha = 0$ (for the proof, see Appendix B of [1]). The segment spacing ΔT is assumed constant. The constraint (2.10b) enforces the continuity of $\mathbf{v}(t)$ between the points $(t_1, t_2, ..., t_{P-1})$. The inner product (2.10d) ensures that $\mathbf{v}(t)$ remains normal to $\mathbf{f}(\mathbf{u}(t), s)$. This constraint arises as a direct result of solving the linear least squares minimisation problem; this is shown in Appendix A of [44]. The derivation of the solution to (2.10) is shown below (see [1] for the full details).

The analytical solution to (2.10c) in each segment can be written as

$$\mathbf{v}(t) = \phi^{t_i,t}\mathbf{v}(t_i) + \int_{t_i}^{t}\phi^{\tau,t}\frac{\partial \mathbf{f}}{\partial s}\,d\tau + \left(\int_{t_i}^{t}\eta(\tau)\,d\tau\right)\mathbf{f}(\mathbf{u}(t), s) \qquad t_i \leq t < t_{i+1} \tag{2.11}$$

where $\phi^{\tau,t}$ is the state transition matrix that satisfies

$$\frac{d\phi^{\tau,t}}{dt} = \left(\left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_t\right)\phi^{\tau,t} \tag{2.12}$$

To proceed, the variable $\eta(\tau)$ is eliminated from (2.11). To do so, (2.11) is first written as

$$\mathbf{v}(t) = \mathbf{v}'(t) + \left(\int_{t_i}^{t}\eta(\tau)\,d\tau\right)\mathbf{f}(\mathbf{u}(t), s) \qquad t_i \leq t < t_{i+1} \tag{2.13}$$

where

$$\mathbf{v}'(t) = \phi^{t_i,t}\mathbf{v}(t_i) + \int_{t_i}^{t} \phi^{\tau,t}\frac{\partial \mathbf{f}}{\partial s}\, d\tau \qquad t_i \leq t < t_{i+1} \tag{2.14}$$

which is the analytical solution of

$$\frac{d\mathbf{v}'}{dt} - \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\mathbf{v}' - \frac{\partial \mathbf{f}}{\partial s} = 0 \tag{2.15}$$

Using (2.10d) and (2.13), an expression for $\eta(\tau)$ can be derived:

$$\int_{t_i}^{t} \eta(\tau)\, d\tau = -\frac{\mathbf{f}(\mathbf{u}(t), s)^T\mathbf{v}'(t)}{\|\mathbf{f}(\mathbf{u}(t), s)\|_2^2} \tag{2.16}$$

Therefore, from (2.13) and (2.14), $\mathbf{v}(t)$ can be written as

$$\mathbf{v}(t) = P_t\mathbf{v}'(t) = \mathbf{v}'(t) - \frac{\mathbf{f}(\mathbf{u}(t), s)^T\mathbf{v}'(t)}{\|\mathbf{f}(\mathbf{u}(t), s)\|_2^2}\mathbf{f}(\mathbf{u}(t), s) \qquad t_i \leq t < t_{i+1} \tag{2.17}$$

where P_t is an $N \times N$ projection operator defined by

$$P_t = I - \frac{\mathbf{f}(t)\mathbf{f}(t)^T}{\mathbf{f}(t)^T\mathbf{f}(t)} \tag{2.18}$$

Using the definition of $\mathbf{v}'(t)$ (2.14) and P_t (2.18), the solution to $\mathbf{v}(t)$ (2.17) is written as

$$\mathbf{v}(t) = P_t\phi^{t_i,t}\mathbf{v}(t_i) + P_t\int_{t_i}^{t} \phi^{\tau,t}\frac{\partial \mathbf{f}}{\partial s}\, d\tau, \qquad t_i \leq t < t_{i+1} \tag{2.19}$$

Finally, using (2.19), the MSS problem (2.10) is reformulated as follows:

$$\underset{\mathbf{v}_i}{\text{Minimise}} \;\; \frac{1}{2}\sum_{i=0}^{P} \|\mathbf{v}_i\|_2^2 \tag{2.20a}$$

$$\text{subject to} \;\; \mathbf{v}_{i+1} = \Phi_{i+1}\mathbf{v}_i + \mathbf{b}_{i+1} \tag{2.20b}$$

where $\mathbf{v}_i = \mathbf{v}(t_i^+)$, $\Phi_{i+1} = P_{t_{i+1}}\phi^{t_i,t_{i+1}}$, and $\mathbf{b}_{i+1} = P_{t_{i+1}}\int_{t_i}^{t_{i+1}}(\phi^{\tau,t_{i+1}})^{\partial \mathbf{f}}/_{\partial s}\, d\tau$. Equation (2.20b) satisfies the original constraints (2.10 b,c,d). The system (2.20) can be written in matrix form as

$$\underset{\mathbf{v}_i}{\text{Minimise}} \;\; \frac{1}{2}\sum_{i=0}^{P}\|\mathbf{v}_i\|_2^2 \tag{2.21a}$$

$$\text{subject to} \;\; A\underline{\mathbf{v}} = \underline{\mathbf{b}} \tag{2.21b}$$

where

$$A = \begin{bmatrix} -\Phi_1 & I & & & \\ & -\Phi_2 & I & & \\ & & \ddots & \ddots & \\ & & & -\Phi_P & I \end{bmatrix} \quad \underline{\mathbf{v}} = \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_P \end{bmatrix} \quad \underline{\mathbf{b}} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_P \end{bmatrix} \tag{2.22}$$

The matrix A is size $NP \times N(P+1)$, while $\underline{\mathbf{v}}$ and $\underline{\mathbf{b}}$ are vectors of length $N(P+1)$ and NP, respectively. According to (2.21), a minimum Euclidean norm solution (2.21a) satisfying (2.21b) is sought. This is a well known least squares problem for under-determined systems in linear algebra [47]. The solution to (2.21) is found by introducing a set of discrete adjoint variables $\underline{\mathbf{w}} = \begin{bmatrix} \mathbf{w}_1^T & \mathbf{w}_2^T & \dots & \mathbf{w}_P^T \end{bmatrix}^T$ and deriving an optimality saddle point system

$$\begin{bmatrix} -I & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \underline{\mathbf{v}} \\ \underline{\mathbf{w}} \end{bmatrix} = \begin{bmatrix} 0 \\ \underline{\mathbf{b}} \end{bmatrix} \tag{2.23}$$

The matrix in the above equation is symmetric and indefinite. The Schur complement system of (2.23) is,

$$S\underline{\mathbf{w}} = \begin{bmatrix} \Phi_1\Phi_1^T + I & -\Phi_2^T & & \\ -\Phi_2 & \Phi_2\Phi_2^T + I & -\Phi_3^T & \\ & \ddots & \ddots & \ddots \\ & & -\Phi_P & \Phi_P\Phi_P^T + I \end{bmatrix} \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_P \end{bmatrix} = \underline{\mathbf{b}} \tag{2.24}$$

where $S = AA^T$. The matrix S is block tri-diagonal, symmetric and positive definite, with size $NP \times NP$. All eigenvalues of S are real and positive. Once the solution $\underline{\mathbf{w}}$ is found, equation (2.23) is used to obtain $\underline{\mathbf{v}}$. Both approaches yield identical results, however, the structure of S can be better exploited to introduce efficient preconditioning (see Chapter 3), therefore the focus of this chapter and the next will be on solving (2.24) rather than (2.23).

Equation (2.24) can be solved iteratively by supplying the matrix-vector (MATVEC) products $S\underline{\mathbf{z}}^{(m)}$ at each iteration m ($\underline{\mathbf{z}}^{(m)}$ is an arbitrary vector), to a Krylov subspace solver, such as Conjugate Gradient (CG) or GMRES. The solver used throughout this thesis was GMRES (without restart), since it offered the fastest convergence with only moderate storage requirements. Other memory efficient solvers such as CG or MINRES should be used if the storage requirements of GMRES become too large. All MATVEC products can be computed by calling a time-stepper for the tangent (or adjoint) equations, with the appropriate initial (or terminal) conditions. For example, the product Φ_i with an arbitrary vector $\mathbf{z}_{i-1}$, requires forward integration of the homogeneous form of (2.15),

$$\frac{d\mathbf{v}'}{dt} - \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\mathbf{v}' = 0 \tag{2.25}$$

with the initial condition $\mathbf{v}'(t_{i-1}^{+}) = \mathbf{z}_{i-1}$ until $t = t_i$. The projection operation (2.18) is then applied, yielding $P_{t_i}\mathbf{v}'(t_i^{-}) = \Phi_i\mathbf{z}_{i-1}$. Similarly, the product of the transpose matrix Φ_i^T with an arbitrary $\mathbf{z}_i$ requires integration of the homogeneous adjoint equation,

$$\frac{d\mathbf{w}}{dt} + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}^T \mathbf{w} = 0 \tag{2.26}$$

backward in time with the terminal condition $\mathbf{w}(t_i^{-}) = P_{t_i}\mathbf{z}_i$ until $t = t_{i-1}$. The product is then given by $\Phi_i^T\mathbf{z}_i = \mathbf{w}(t_{i-1}^{+})$. The iterative solution to (2.24) dominates the computational cost of the tangent MSS algorithm. Each iteration requires the integration of (2.25) and (2.26) across all segments, which can be performed in parallel.

The sensitivity $d\overline{J^{(T)}}/ds$ is obtained by integrating

$$\frac{d\overline{J^{(T)}}}{ds} = \frac{1}{T}\sum_{i=0}^{P-1}\int_{t_i}^{t_{i+1}}\left(\frac{\partial J}{\partial \mathbf{u}}^T \mathbf{v}'\right) dt + \frac{1}{T}\left(\sum_{i=0}^{P-1}\frac{(\mathbf{f}_{i+1})^T\mathbf{v}'(t_{i+1})}{\|\mathbf{f}_{i+1}\|_2^2}(\overline{J^{(T)}} - J_{i+1})\right) + \frac{\partial \overline{J^{(T)}}}{\partial s} \tag{2.27}$$

The full process to compute $d\overline{J^{(T)}}/ds$ is summarised in Algorithms (1) and (2).

Algorithm 1: Tangent MSS Solver

---

1. Integrate $d\mathbf{u}/dt = \mathbf{f}(\mathbf{u}, s)$ in $[0, T]$ to find $\mathbf{u}(t)$. Compute $\mathbf{f}_i = \mathbf{f}(t_i)$.
2. Compute the projections $P_{t_{i+1}}$ $(i = 0, 1, ..., P-1)$ using (2.18).
3. Form the vector $\underline{\mathbf{b}}$ (RHS of equation 2.24). This is done by integrating (2.15) in all segments $(1, 2, ..., P)$ with the initial conditions $\mathbf{v}'(t_{i-1}^+) = 0$. Then form $\underline{\mathbf{b}} = \left[\left(P_{t_1}\mathbf{v}'(t_1^-)\right)^T \quad \left(P_{t_2}\mathbf{v}'(t_2^-)\right)^T \quad ... \quad \left(P_{t_P}\mathbf{v}'(t_P^-)\right)^T\right]^T$.
4. Solve the Schur complement system (2.24) using a suitable iterative solver that accepts MATVEC products. Use Algorithm (2) (see below) to compute the MATVEC product at each iteration.
5. Compute $\underline{\mathbf{v}}$ from (2.23), then integrate (2.15) in all segments to find $\mathbf{v}'(t)$.
6. Find $d\overline{J^{(T)}}/ds$ by evaluating (2.27).

Algorithm 2: Schur MATVEC

Compute the MATVEC product $S\underline{\mathbf{z}} = AA^T\underline{\mathbf{z}}$ for an arbitrary vector $\underline{\mathbf{z}}$:

i First compute $\underline{\mathbf{v}} = A^T\underline{\mathbf{z}}$: Integrate (2.26) backward in time in all segments $(i = 1, 2, ..., P)$ with the terminal conditions $\mathbf{w}(t_i^-) = P_{t_i}\mathbf{z}_i$. This gives $\mathbf{w}(t_{i-1}^+) = \Phi_i^T\mathbf{z}_i$. Form $\mathbf{v}_{i-1} = \mathbf{z}_{i-1} - \mathbf{w}(t_{i-1}^+)$ (for $i = 2, 3, ..., P-1$). Note that $\mathbf{v}_0 = -\mathbf{w}(t_0^+)$ and $\mathbf{v}_P = \mathbf{z}_P$.

ii Next compute $S\underline{\mathbf{z}} = A\underline{\mathbf{v}}$: Integrate (2.25) forward in time in all segments with the initial conditions $\mathbf{v}'(t_{i-1}^+) = \mathbf{v}_{i-1}$ to obtain $\mathbf{v}'(t_i^-)$. Apply the projection operators to find $P_{t_i}\mathbf{v}'(t_i^-) = \Phi_i\mathbf{v}_{i-1}$. Form $S\underline{\mathbf{z}} = \left[\left(-P_{t_1}\mathbf{v}'(t_1^-) + \mathbf{v}_1\right)^T \quad \dots \quad \left(-P_{t_P}\mathbf{v}'(t_P^-) + \mathbf{v}_P\right)^T\right]^T$.

2.4.2 Adjoint MSS

The tangent MSS method (Algorithm 1) is not useful when the sensitivities to many parameters are required. To derive the adjoint version of MSS that computes sensitivities to many parameters simultaneously, (2.27) is combined with (2.23) as follows [1]:

$$\frac{d\overline{J^{(T)}}}{ds} = \left(\begin{bmatrix} \underline{\mathbf{g}}^T & 0 \end{bmatrix} \begin{bmatrix} \underline{\mathbf{v}} \\ \underline{\mathbf{w}} \end{bmatrix} + h + \frac{\partial \overline{J^{(T)}}}{\partial s} \right) + \begin{bmatrix} \underline{\hat{\mathbf{v}}}^T & \underline{\hat{\mathbf{w}}}^T \end{bmatrix} \left[\begin{bmatrix} -I & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \underline{\mathbf{v}} \\ \underline{\mathbf{w}} \end{bmatrix} - \begin{bmatrix} 0 \\ \underline{\mathbf{b}} \end{bmatrix} \right] \tag{2.28}$$

where the terms in the curved brackets are identical to the right-hand-side of (2.27) and the last term is the residual of (2.23), constrained with a new set of adjoint variables, $\underline{\hat{\mathbf{v}}} = \begin{bmatrix} \hat{\mathbf{v}}_1^T & \hat{\mathbf{v}}_2^T & \dots & \hat{\mathbf{v}}_P^T \end{bmatrix}^T$ and $\underline{\hat{\mathbf{w}}} = \begin{bmatrix} \hat{\mathbf{w}}_1^T & \hat{\mathbf{w}}_2^T & \dots & \hat{\mathbf{w}}_P^T \end{bmatrix}^T$. The vector $\underline{\mathbf{g}}^T = \begin{bmatrix} \mathbf{g}_1^T & \mathbf{g}_2^T & \dots & \mathbf{g}_P^T & 0 \end{bmatrix}^T$, where

$$\mathbf{g}_i^T = \frac{1}{T} \int_{t_{i-1}}^{t_i} \left(\frac{\partial J}{\partial \mathbf{u}}^T \phi^{t,t_i} \right) dt + \frac{1}{T} (\overline{J^{(T)}} - J_i) \frac{\mathbf{f}_i^T \phi^{t_{i-1},t_i}}{\|\mathbf{f}_i\|_2^2} \tag{2.29}$$

and h is a scalar. Equation (2.28) is rearranged (see Appendix D in [1] for the full details) to arrive at the new adjoint system,

$$\begin{bmatrix} -I & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \underline{\hat{\mathbf{v}}} \\ \underline{\hat{\mathbf{w}}} \end{bmatrix} = \begin{bmatrix} -\underline{\mathbf{g}} \\ 0 \end{bmatrix} \tag{2.30}$$

which has the Schur complement system,

$$S\underline{\hat{\mathbf{w}}} = -A\underline{\mathbf{g}} \tag{2.31}$$

where $S = AA^T$ is identical to the tangent MSS Schur complement matrix (2.24). Within each time segment ($i = 1, 2, \dots, P$), the continuous time adjoint variables

$\hat{\mathbf{w}}(t) = \begin{bmatrix} \hat{w}_1(t) & \hat{w}_2(t) & \dots & \hat{w}_N(t) \end{bmatrix}^T$ must satisfy

$$\frac{d\hat{\mathbf{w}}}{dt} = -\frac{\partial \mathbf{f}}{\partial \mathbf{u}}^T \hat{\mathbf{w}} - \frac{1}{T}\frac{\partial J}{\partial \mathbf{u}} \qquad t_{i-1} \leq t < t_i \tag{2.32}$$

which is integrated backward in time in all segments with the terminal conditions,

$$\hat{\mathbf{w}}(t_i) = P_{t_i}\hat{\mathbf{w}}_i + \frac{1}{T}\frac{\overline{J^{(T)}} - J_i}{\mathbf{f}_i^T \mathbf{f}_i}\mathbf{f}_i \tag{2.33}$$

where $\hat{\mathbf{w}}_i$ is obtained from the solution to (2.31). Finally, using $\hat{\mathbf{w}}(t)$, the sensitivity of $\overline{J^{(T)}}$ to any parameter can be found by evaluating the integral

$$\frac{d\overline{J^{(T)}}}{ds} = \left(\sum_{i=1}^{P} \int_{t_{i-1}}^{t_i} \frac{\partial \mathbf{f}}{\partial s}^T \hat{\mathbf{w}}(t)\, dt \right) + \frac{1}{T}\int_0^T \frac{\partial J}{\partial s}\, dt \tag{2.34}$$

Similar to the tangent MSS method, the solution to the adjoint Schur complement system (2.31) dominates the computational cost of the adjoint method. The Schur complement matrices S for both systems are identical; hence the cost of solving (2.24) or (2.31) is similar. However, as will be shown in Chapter (3), solving (2.24) is computationally expensive even for low dimensional turbulent systems. The aim of next chapter is to significantly reduce the computational cost by introducing efficient preconditioning.

Chapter 3

Convergence Acceleration of the MSS Algorithm[1]

3.1 Introduction

The convergence rate of iterative Krylov subspace solvers for symmetric, positive definite matrices like S (defined in equation 2.24) depends on the distribution of the matrix eigenvalues [49]. For such systems, the eigenvalues are all positive and real. If all of them are tightly clustered around a few points away from the origin, then fast convergence would be expected. On the other hand, widely spread eigenvalues without tight clustering can lead to slow convergence. The objective of a preconditioner is to reduce the spread of the eigenvalues, i.e. to reduce the condition number, $\kappa(S) = {}^{\mu_{max}(S)}/_{\mu_{min}(S)}$, where $\mu_{max}(S)$ and $\mu_{min}(S)$ are the maximum and minimum eigenvalues of S, respectively.

The matrix S has large κ, making convergence slow, even for low dimensional sys-

[1]Results from this chapter have appeared in [48]: K. Shawki and G. Papadakis. A preconditioned Multiple Shooting Shadowing algorithm for the sensitivity analysis of chaotic systems. Journal of Computational Physics, 398, 2019. https://doi.org/10.1016/j.jcp.2019.108861

tems. Small eigenvalues ($0 < \mu(S) \ll 1$) are present, owing to the lack of strict uniform hyperbolicity [31], while large eigenvalues $\mu(S) \gg 1$ are due to the exponential growth of $\mathbf{v}(t)$ within a segment. To allow MSS to be applied efficiently to high dimensional systems, a preconditioner is proposed in this chapter that can accelerate the convergence rate by many orders of magnitude. Since the tangent and adjoint MSS systems (equations (2.24) and (2.31) respectively) have identical matrices S, their convergence rates with linear solvers are very similar, and the same preconditioner may be applied to either system. For the numerical demonstrations that follow, the preconditioner derived in this chapter is applied to the tangent system (2.24).

This chapter is structured as follows: Section (3.2) provides a literature review of preconditioners relevant to the KKT and Schur complement matrices. In Section (3.3), a preconditioner based on partial singular value decomposition is derived. A simplified version is proposed in Section (3.4), and both preconditioners are applied to the Lorenz system and the Kuramoto Sivashinsky equation in Section (3.5). In Section (3.6), an important finding on the effect of the smallest eigenvalues of S on the accuracy of the computed sensitivities is presented. Regularisation of the preconditioned system is considered in Section (3.7), while the computational costs of the iterative system solutions (with and without preconditioning) are compared in Section (3.8).

3.2 A Review of Preconditioners

An extensive survey of preconditioners for saddle point problems, such as the tangent or adjoint KKT systems (equations (2.23) and (2.30) respectively), is available in [50]. Preconditioners can be applied to the 2×2 block systems (equation 2.23 or 2.30) or directly to the Schur complement systems (equation 2.24 or 2.31). It is important to note that in either case, an easily invertible approximation of S is required. For the present case, there is an additional restriction, namely that the matrix S is not explicitly known, only its product with a vector can be computed. The preconditioner should also be matrix-free and must be computed using matrix-vector products (computing and storing S is out of the question for long trajectories and large N).

Efficient approximations of S can be made if one takes into account the structure of the problem. For example, in the finite element solution of the incompressible Navier-Stokes equations, where pressure plays the role of the Lagrange multiplier enforcing the incompressibility condition, the Schur complement can be interpreted as a discretisation of a second-order diffusion operator. This is not surprising, because pressure is governed by a Poisson equation. For this operator, the action of S^{-1} can be efficiently approximated by a multigrid iteration (more details can be found in [51, 52]). For the present case, a second-order partial differential equation for the adjoint variable $\mathbf{w}(t)$ was derived in [44] from the system (2.4). However, this equation is too complex to solve, and it is not evident how it can be simplified in order to aid the construction of a preconditioner.

Recently, McDonald et al. [53] proposed a block circulant preconditioner for the 'all-at-once' evolution of a linear system of ODEs with constant coefficients. Here 'all-at-once' means that the space/time problem is written as a monolithic linear system. This problem is very similar to the present case. The authors exploit the block Toeplitz structure of the system to develop an efficient preconditioner that results in the number of Krylov subspace iterations being independent of the number of time-steps. Unfortunately, this approach is also unsuitable for the current problem. The reason is that the block Toeplitz structure of the matrix in [53] is a direct consequence of the fact that the coefficients of the ODE are constant. For the current problem, the blocks Φ_i of the matrix A in equation (2.22) depend on time

t, so this approach cannot be applied.

In the course of this work, several preconditioners which have been proposed in the literature for the solution of general saddle point systems were investigated. The preconditioners were applied using matrix-vector products only, but did not exploit the properties of the underlying physical problem. For example, Cao et al. [54] proposed a splitting of the block matrix (2.23) into two matrices, one of which is used as a left preconditioner. The splitting contains an adjustable parameter, and theoretical analysis shows that the largest eigenvalue of the preconditioned system is one (independent of the value of the adjustable parameter), provided that the preconditioner is applied exactly. In practice, however, this cannot be achieved, and the preconditioner had to be applied approximately. Although the method relies on matrix-vector products only, it did not reduce the overall cost. The preconditioner of Golub et al. [55] guarantees that the eigenvalues remain bounded within two intervals, $[-1, (1-\sqrt{5})/2]$ and $[1, (1+\sqrt{5})/2]$ when applied exactly. This tight clustering ensures fast convergence but also needs to be approximated. Unfortunately, this approach did not reduce the overall cost as well. Standard preconditioners for general matrices, such as incomplete LU decomposition [49], are also not suitable and they additionally require storage of the matrix.

A new preconditioner that can be computed using matrix-vector products only, has moderate storage requirements, and that exploits the properties of the matrix A (equation 2.22), is therefore required. The central idea is to identify the fastest growing (singular) modes of the matrix A and annihilate them. These modes are partially responsible for the large condition number $\kappa(S)$ (the other contributor to the large κ is described in Section (3.7) and is dealt with differently), and must be annihilated for fast convergence. In this way, it is expected that it would be easier to minimise the norm (2.21a) while preserving the continuity of $\mathbf{v}(t)$ across segments. This can be achieved using partial singular value decomposition.

3.3 Preconditioning Based on Partial Singular Value Decomposition

Recall the Schur complement system (2.24), $AA^T\underline{\mathbf{w}} = \underline{\mathbf{b}}$. The singular value decomposition (SVD) of A reads

$$A = U\Sigma V^T \tag{3.1}$$

where U is a $NP \times NP$ unitary matrix, Σ is a $NP \times N(P+1)$ quasi-diagonal matrix, and V is a $N(P+1) \times N(P+1)$ also unitary matrix. The columns of U contain the left singular vectors of A, while the right singular vectors of A make up the columns of V. Σ contains the singular values of A, which are denoted by $\sigma(A)$, in the $NP \times NP$ diagonal sub-matrix (the last N columns consist of zeros, which are ignored). The singular values are ordered from largest to smallest, i.e. the diagonal elements are $\Sigma_{ii} = \sigma_i \quad (i = 1, \ldots, NP)$ and $\sigma_1 > \sigma_2 > \ldots > \sigma_{NP}$. Note that $\sigma(A) = \sqrt{\mu(S)}$, where $\mu(S)$ are the eigenvalues of S. Using the fact that $V^TV = I$ (since the columns of V are orthonormal), the matrix S can be written as

$$S = AA^T = U\Sigma\Sigma^T U^T = U\Sigma^2 U^T \tag{3.2}$$

and since $U^TU = I$, this can be easily inverted to give

$$S^{-1} = U\Sigma^{-2}U^T \tag{3.3}$$

Equation (3.3) forms an exact preconditioner, which of course is not practical to compute. However, an approximate preconditioner can be formed using the leading l singular modes only, i.e. by performing a partial SVD. S can be written as

$$S = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1^2 & 0 \\ 0 & \Sigma_2^2 \end{bmatrix} \begin{bmatrix} U_1^T \\ U_2^T \end{bmatrix} = U_1\Sigma_1^2U_1^T + U_2\Sigma_2^2U_2^T \tag{3.4}$$

where U is partitioned as $U = \begin{bmatrix} U_1 & U_2 \end{bmatrix}$, with $U_1 = \begin{bmatrix} u_1 & u_2 & ... & u_l \end{bmatrix}$ and $U_2 = \begin{bmatrix} u_{l+1} & u_{l+2} & ... & u_{NP} \end{bmatrix}$. Matrix $\Sigma_1^2 = diag(\sigma_1^2, \sigma_2^2, ..., \sigma_l^2)$ contains the singular values corresponding to U_1, and $\Sigma_2^2 = diag(\sigma_{l+1}^2, \sigma_{l+2}^2, ..., \sigma_{NP}^2)$ contains the rest. By replacing the diagonal submatrix Σ_2^2 with the identity matrix $I_2 = I_{(NP-l)}$, an approximation $\hat{S}_{(l)}$,

$$\hat{S}_{(l)} = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1^2 & 0 \\ 0 & I_2 \end{bmatrix} \begin{bmatrix} U_1^T \\ U_2^T \end{bmatrix} = U_1 \Sigma_1^2 U_1^T + U_2 U_2^T \tag{3.5}$$

is obtained, which can be easily inverted to give

$$\hat{S}_{(l)}^{-1} = M_{(l)} = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1^{-2} & 0 \\ 0 & I_2 \end{bmatrix} \begin{bmatrix} U_1^T \\ U_2^T \end{bmatrix} = U_1 \Sigma_1^{-2} U_1^T + U_2 U_2^T \tag{3.6}$$

The product $M_{(l)}S$ now becomes

$$M_{(l)}S = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & \Sigma_2^2 \end{bmatrix} \begin{bmatrix} U_1^T \\ U_2^T \end{bmatrix} \tag{3.7}$$

which indicates that $M_{(l)}$ has deflated the l largest singular values of matrix S to 1, while leaving the rest unaltered. To avoid computing the columns of U_2 in (3.6), the orthogonality relation $U_2 U_2^T = I - U_1 U_1^T$ is invoked, leading to

$$M_{(l)} = U_1 \Sigma_1^{-2} U_1^T + (I - U_1 U_1^T) \tag{3.8}$$

Therefore in order to form $M_{(l)}$, only Σ_1 and U_1 are needed. $M_{(l)}$ is deployed as a left preconditioner to convert (2.24) into a better conditioned system of the form

$$M_{(l)} S \underline{\mathbf{w}} = M_{(l)} \underline{\mathbf{b}} \tag{3.9}$$

which is solved for $\underline{\mathbf{w}}$.

Preconditioning based on partial singular value decomposition has been applied in the past, for example in [56] to solve ill-conditioned least squares problems arising in image de-blurring applications. A preconditioner based on the partial Krylov-Schur decomposition of a matrix (see [57] for details) was also used in [58] to accelerate the computation of limit cycles for thermoacoustic systems. The preconditioner had a form very similar to (3.8), the difference being that instead of the diagonal matrix Σ_1^{-2} in the first term on the right-hand side, the inverse of an upper triangular $l \times l$ matrix was taken. The preconditioner reduced the condition number significantly, speeded up the convergence of GMRES, but lead to modest overall cost savings (mainly due to the cost of converging the eigenvalues used to form the preconditioner). Sánchez and Net [59] also used a similar preconditioner for accelerating the convergence of a multiple shooting algorithm for finding periodic orbits.

The Lanczos bidiagonalization algorithm [60] (implemented in the 'svds' command of MATLAB) can be employed to form an approximation to $M_{(l)}$ using q iterations. It is important to note that the algorithm requires matrix-vector products only, making it suitable for the current problem. The approximation of $M_{(l)}$ after q iterations of the algorithm is

$$M_{(l)}^{(q)} = U_1^{(q)} \Sigma_1^{-2(q)} U_1^{T(q)} + (I - U_1^{(q)} U_1^{T(q)}) \tag{3.10}$$

Computing the approximations $U_1^{(q)}$ and $\Sigma_1^{(q)}$ using the function 'svds' of MATLAB requires the evaluation of a large number of matrix-vector products $A\underline{\mathbf{x}}$ and $A^T\underline{\mathbf{z}}$, where $\underline{\mathbf{x}}$ and $\underline{\mathbf{z}}$ are algorithm-generated vectors of length $N(P+1)$ and NP, respectively. More specifically, each iteration requires at least $l+2$ (the smallest permissible Krylov subspace dimension) applications of A and A^T. Even with one or two iterations of the algorithm, the cost of computing $M_{(l)}^{(q)}$ would most likely outweigh the cost savings due to solving the preconditioned system (3.9). A much cheaper alternative is therefore proposed in the next section.

3.4 A Simplified Block Diagonal Preconditioner

A much cheaper preconditioner can be formed by considering the approximation $\tilde{A}$,

$$\tilde{A} = \begin{bmatrix} -\Phi_1 & 0 & & & \\ & -\Phi_2 & 0 & & \\ & & \ddots & \ddots & \\ & & & -\Phi_P & 0 \end{bmatrix} \tag{3.11}$$

which is obtained from A by neglecting the upper diagonal. The corresponding Schur complement matrix $\tilde{S} = \tilde{A}\tilde{A}^T$ is a block diagonal matrix that takes the form

$$\tilde{S} = \begin{bmatrix} \Phi_1\Phi_1^T & & & \\ & \Phi_2\Phi_2^T & & \\ & & \ddots & \\ & & & \Phi_P\Phi_P^T \end{bmatrix} \tag{3.12}$$

This form indicates that each segment is now decoupled from the neighbouring segments on either side. A block diagonal preconditioner (BDP) M_{BD} that approximates the inverse $\tilde{S}^{-1}$, can be constructed, i.e.,

$$\mathrm{M}_{BD} \approx \tilde{S}^{-1} = \begin{bmatrix} (\Phi_1\Phi_1^T)^{-1} & & & \\ & (\Phi_2\Phi_2^T)^{-1} & & \\ & & \ddots & \\ & & & (\Phi_P\Phi_P^T)^{-1} \end{bmatrix} \tag{3.13}$$

Each diagonal block in (3.13) can be approximated as before using partial singular value decompositions, i.e.,

$$\mathrm{M_{BD}}_{(l)}^{(q)} = diag(\mathrm{M}_{(l),1}^{(q)}, \mathrm{M}_{(l),2}^{(q)}, ..., \mathrm{M}_{(l),P}^{(q)}) \tag{3.14a}$$

$$\mathrm{M}_{(l),i}^{(q)} = \underline{U}_{1,i}\underline{\Sigma}_{1,i}^{-2}\underline{U}_{1,i}^{T} + (I - \underline{U}_{1,i}\underline{U}_{1,i}^{T}) \tag{3.14b}$$

where i is the segment number, q is the number of iterations and l is the number of retained singular modes in each segment. The matrices $\underline{\Sigma}_{1,i}$ and $\underline{U}_{1,i}$ correspond to the l leading singular values and the left singular vectors of Φ_i, respectively. The superscript (q) and subscript (l) have been removed from $\underline{\Sigma}_{1,i}$ and $\underline{U}_{1,i}$ for clarity.

The preconditioner $\mathrm{M_{BD}}_{(l)}^{(q)}$ has several advantages. Firstly, it is much cheaper to construct than $M_{(l)}^{(q)}$; in fact, it is $O(P)$ times cheaper. To see why, recall that computing $M_{(Pl)}$ using Lanczos bidiagonalisation requires at least $Pl + 2$ matrix-vector products with A and A^T. This deflates the leading Pl eigenvalues to one. Preconditioning with $\mathrm{M_{BD}}_{(l)}$ is similar to preconditioning with $M_{(Pl)}$ in terms of the eigenvalue deflation. However, computing $\mathrm{M_{BD}}_{(l)}$ only requires $l+2$ matrix-vector products with Φ_i and Φ_i^T per segment (i.e. the equivalent of $l+2$ applications of A and A^T). Secondly, the computation of $\mathrm{M_{BD}}_{(l)}^{(q)}$ is fully parallelisable in time. Each processor can be assigned to one segment and computations can proceed independently, because each segment is treated separately, i.e. there is no message passing between processors. On the other hand, computing $M_{(Pl)}$ requires passing of all the degrees of freedom of the computational domain from the processor operating at segment $i+1$, to the processor operating at segment i (due to the off-diagonal identities of the matrix A). This can be overlapped with the computation at segment i, but it is not clear a priori that message passing can be completed before the end of the computation (this is probably case dependent). Thirdly, storage costs with respect to $M_{(Pl)}$ are reduced by a factor of P (for $\mathrm{M_{BD}}_{(l)}$, there are Pl vectors which need to be stored, each with length N, whereas $M_{(Pl)}$ requires the storage of Pl

vectors of length PN each).

Both S and $\mathrm{M_{BD}}_{(l)}^{(q)}$ are symmetric, positive definite matrices, making the conjugate gradient method applicable. $\mathrm{M_{BD}}_{(l)}^{(q)}$ is used as a left preconditioner for the original system, i.e. the system

$$\mathrm{M_{BD}}_{(l)}^{(q)} S \underline{\mathbf{w}} = \mathrm{M_{BD}}_{(l)}^{(q)} \underline{\mathbf{b}} \tag{3.15}$$

is solved for $\underline{\mathbf{w}}$. In the following section, the condition number of the matrix $\mathrm{M_{BD}}_{(l)}^{(q)} S$, and the performance of the preconditioner for two standard problems (the Lorenz system and the Kuramoto Sivashinsky equation) are investigated.

3.5 Numerical Examples

3.5.1 The Lorenz System

The well-known Lorenz system takes the form

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x) \\ \frac{dy}{dt} &= x(\rho - z) - y \\ \frac{dz}{dt} &= xy - \beta z \end{aligned} \tag{3.16}$$

where σ, ρ and β are system parameters. The Lorenz system is a common test case for chaotic sensitivity analysis methods. The parameters $\sigma = 10$ and $\beta = 8/3$ are kept constant, while ρ is varied. The attractor type varies with ρ as follows [61]: for $1 < \rho < 24.74$, there are two stable fixed points at $x = y = \pm\sqrt{\beta(\rho - 1)}$, $z = \rho - 1$, quasi-hyperbolic attractors for $24.06 < \rho < 31$, non-hyperbolic attractors for $31 < \rho < 99.5$ and periodic limit cycles for $\rho > 99.5$. The largest Lyapunov exponent λ_{max} increases from $\lambda_{max} \approx 0.8$ at $\rho = 24$ to $\lambda_{max} \approx 1.7$ at $\rho = 96$ [62] (roughly a linear growth with ρ with frequent dips). The remaining exponents are

$\lambda_2 = 0$ and $\lambda_3 < 0$.

The objective considered is

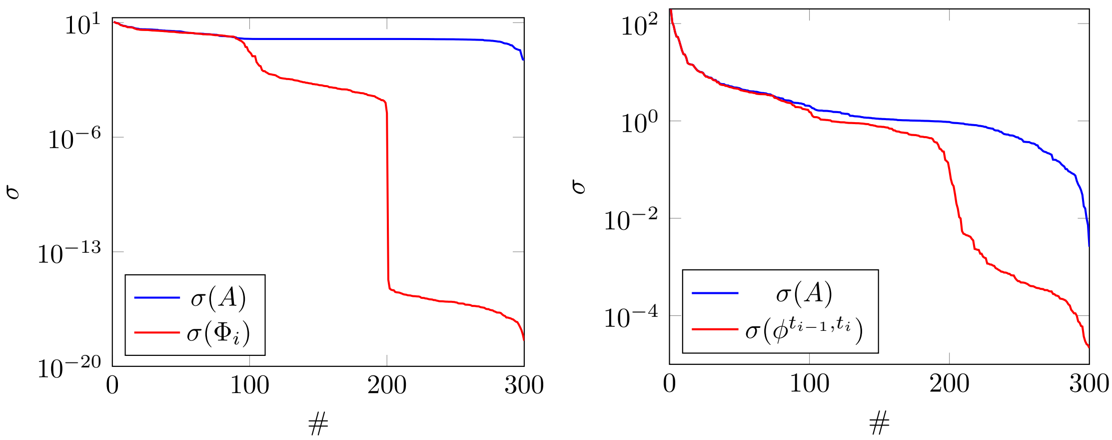
$$\overline{J^{(T)}} = \frac{1}{T}\int_0^T z\,dt \tag{3.17}$$

Using $\sigma = 10$ and $\beta = {}^8\!/_3$, sensitivities were sought with respect to the parameter ρ, i.e. $d\overline{J^{(T)}}/d\rho$. MATLAB's variable-step Runge-Kutta solver (ode45) was used to compute the trajectory $\mathbf{u}_{ref}(t)$ and to perform all tangent and adjoint time-stepping.

Figure (3.1a) shows a comparison of the singular values of A, plotted together with the union of the singular values of all Φ_i, ordered from largest to smallest. There are in total $3P = 300$ singular values for the case examined. Each segment is located at a different place on the attractor, and has 3 local finite-time Lyapunov exponents λ' associated with it. It is well known that λ' can fluctuate significantly around the average values as the trajectory is traced [63]. It can be seen that $\sigma(\Phi_i)$ is close to $\sigma(A)$ for the largest P values, and thereafter the two curves start to deviate. The last P values of $\sigma(\Phi_i)$ are very small, which is expected, since $\Phi_i = P_{t_i}\phi^{t_{i-1},t_i}$ is almost singular due to the projection P_{t_i} (equation 2.18). To remove the effect of the singularity which distorts the comparison, Figure (3.1b) shows $\sigma(A(\phi^{t_{i-1},t_i}))$ and $\sigma(\phi^{t_{i-1},t_i})$, where $A(\phi^{t_{i-1},t_i})$ is evaluated using ϕ^{t_{i-1},t_i}, i.e. without applying the projection P_{t_i} to the state transition matrix. The matching for the first P singular values is now much more clearly seen.

In [1], it was shown that $\mu_{max}(S) = \sigma^2_{max}(A)$ is related to the largest singular value of Φ_i across all segments, $\max\limits_i\left(\sigma_i^{(1)}\right)$, as $\mu_{max}(S) = 1 + \max\limits_i\left(\sigma_i^{(1)}\right)^2$, provided that $\max\limits_i\left(\sigma_i^{(1)}\right)^2 \gg 1$, in which case $\mu_{max}(S) \approx \max\limits_i\left(\sigma_i^{(1)}\right)^2$. Figure (3.1) shows that this approximation holds for many more eigenvalues. For the Lorenz system, it holds for approximately P eigenvalues, each corresponding to the λ'_{max} for each segment.

Figure (3.2) zooms in on the largest P singular values for two different ΔT. The singular values $\sigma(\Phi_i)$ approximate $\sigma(A)$ better for the larger ΔT value. This is

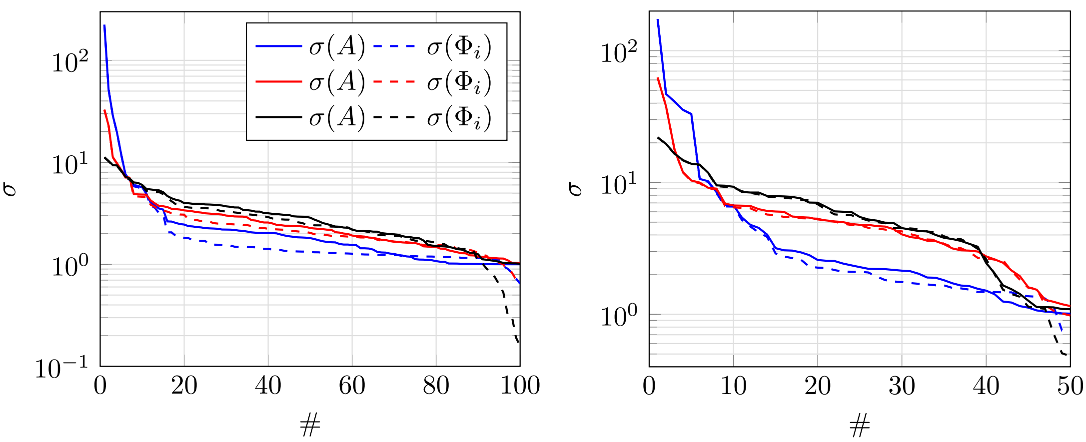


(a) Projections P_{t_i} applied

(b) No projections applied

Figure 3.1: A distribution of $\sigma(A)$, $\sigma(\Phi_i) = \sigma(P_{t_i}\phi^{t_{i-1},t_i})$ and $\sigma(\phi^{t_{i-1},t_i})$, ordered from the largest to the smallest values. Obtained for $\rho = 80$ with $T = 50$ and $\Delta T = 0.5$ ($P = 100$ segments).

because the diagonal blocks of the matrix A become more dominant for $\Delta T = 1$, and the off-diagonal identity matrices can be neglected (refer to equation 3.11), without impairing much the accuracy of the large singular values.

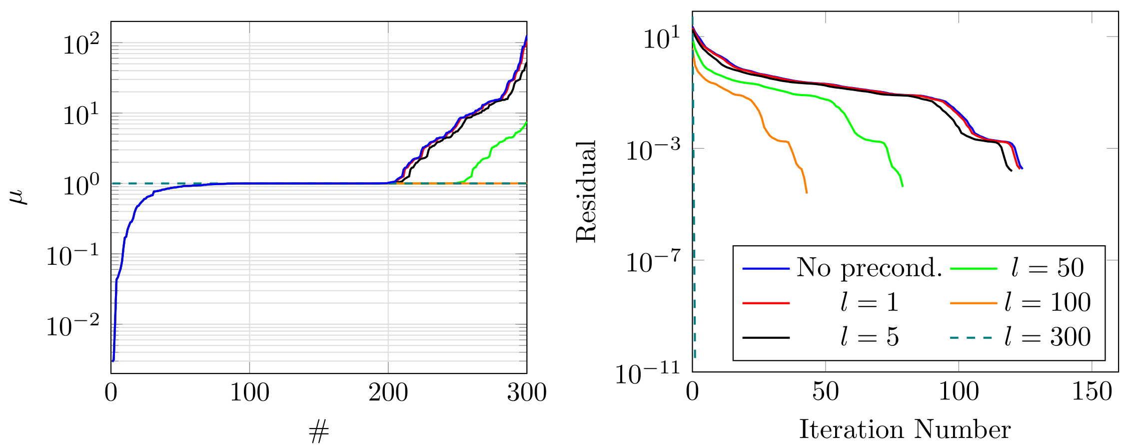


(a) $\Delta T = 0.5$ ($P = 100$ segments)

(b) $\Delta T = 1$ ($P = 50$ segments)

Figure 3.2: A distribution of the largest P values of $\sigma(A)$ and the largest $\sigma(\Phi_i)$ of each segment. Obtained with $T = 50$. Blue: $\rho = 40$, red: $\rho = 60$, black: $\rho = 80$.

Next, the spectrum of the preconditioned system is investigated, starting with the exact preconditioner $M_{(l)}$ (3.8). The l largest singular values and vectors of S used to form $M_{(l)}$ were obtained iteratively until convergence. Figure (3.3a) shows the eigenvalues $\mu(S)$ and $\mu(M_{(l)}S)$. For the case examined, the matrix S has approximately $P = 100$ eigenvalues $\mu(S) > 1$ (blue line), corresponding to the local positive Lyapunov exponent in each segment. Eigenvalues $\mu(S) = 1$ correspond to the stable exponent[1], and eigenvalues $\mu(S) < 1$ correspond to the neutrally stable exponent. Different values of l were used to construct $M_{(l)}$ (the values are reported in Figure 3.3b). It is clear that $M_{(l)}$ effectively deflates the l largest eigenvalues of the preconditioned system $\mu(M_{(l)}S)$ to 1, while leaving the rest unaltered. As l increases, more and more eigenvalues are clustered closer to 1, leading to increasingly faster convergence (shown in Figure 3.3b). When $l = P(= 100)$, $\mu_{max}(M_{(l)}S) = 1$, and convergence still takes about 50 iterations. For the limiting case of $l = 3P(= 300)$, all eigenvalues are equal to 1 (see Panel a) and convergence is achieved in just one iteration, as expected. This indicates that for fast convergence, it is not sufficient only to deflate the large eigenvalues (μ>1), but also to increase (regularise) the very small ($0 < \mu \ll 1$) eigenvalues. Such small eigenvalues make the system more singular, with implications to the accuracy of the computed sensitivity; this is explored later in Section (3.6).



(a) Eigenvalues

(b) Residuals

Figure 3.3: Eigenvalues (ordered from smallest to largest) and convergence residuals for the original system S (blue line) and the preconditioned system $M_{(l)}S$, for different l (number of singular modes). Obtained for $\rho = 80$ with $T = 50$ and $\Delta T = 0.5$ ($P = 100$ segments). The GMRES solver used a relative tolerance of 1×10^{-5}.

[1]Consider a stable system with all negative exponents (i.e. all $\sigma(\Phi_i) \approx 0$). Then all $\mu(S) = 1$ due to the identities on the main diagonal of S (equation 2.24).

The performance of the BDP (3.14) is now investigated. Figure (3.4) shows $\mu(S)$, $\mu(M_{(25)}S)$ and $\mu(\mathrm{M}_{\mathrm{BD}(1)}S)$. The eigenvalues $\mu(S) > 1$ vary significantly in value because they are related to $\lambda'^{(i)}_{max}$, i.e. to the largest local finite time Lyapunov exponent which varies considerably across segments. The number of segments is $P = 25$, and only one singular value has been computed until convergence in each segment (for the BDP). Both preconditioners are aiming to deflate the P largest eigenvalues. It is clear that the matrix $\mathrm{M}_{\mathrm{BD}(1)}S$ has a very similar eigenvalue spectrum compared to $\mu(M_{(25)}S)$. It can be seen that $\mu_{max}(M_{(25)}S) = 1$, while $\mu_{max}(\mathrm{M}_{\mathrm{BD}(1)}S) \approx 2$. Most importantly, $\mathrm{M}_{\mathrm{BD}(1)}S$ has clustered the P largest eigenvalues in the narrow interval $[1, 2]$. This indicates that the approximate BDP preconditioner encapsulates reliable information for the fastest growing modes, which results in the suppression of $\mu_{max}(\mathrm{M}_{\mathrm{BD}(1)}S)$ by four orders of magnitude. However, there is a slight reduction in $\mu_{min}(\mathrm{M}_{\mathrm{BD}(1)}S)$ with respect to $\mu_{min}(S)$ and $\mu_{min}(M_{(25)}S)$. This is likely because $\mathrm{M}_{\mathrm{BD}(1)}$ is only an approximation to $M_{(25)}$. Note that $\mu_{min}(M_{(25)}S) = \mu_{min}(S)$, as expected, according to equation (3.7).

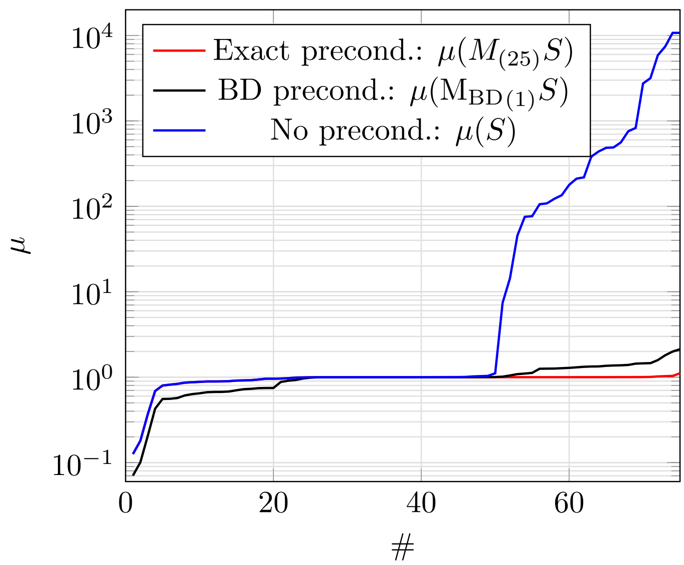


Figure 3.4: Eigenvalues of the original and preconditioned systems for $T = 50$, $\Delta T = 2$ ($P = 25$) and $\rho = 80$.

Figure (3.5) shows the convergence rates for different values of ρ. As expected, using the exact preconditioner $M_{(l)}$ provides considerably faster convergence compared to

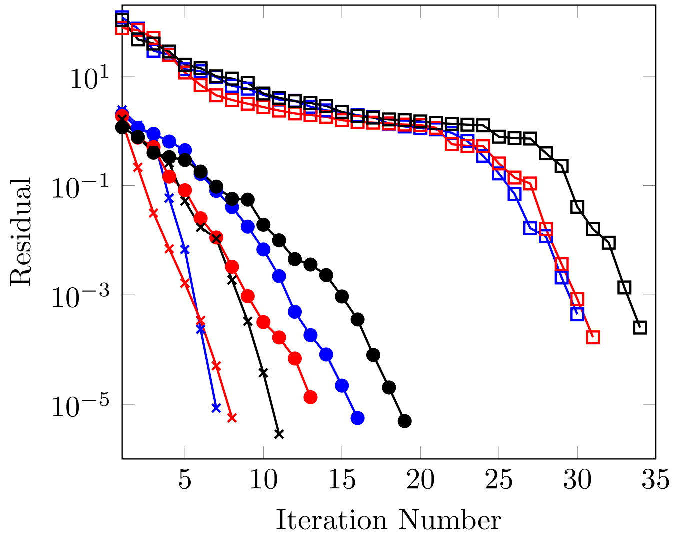


Figure 3.5: Convergence history for the original and preconditioned systems using $T = 50$ and $\Delta T = 2$ ($P = 25$). Blue: $\rho = 40$, red: $\rho = 60$, black: $\rho = 80$. Squares: S, crosses: $M_{(25)}S$, circles: $\mathrm{M}_{\mathrm{BD}(1)}S$. The GMRES solver used a relative tolerance of 1×10^{-5}.

the original system (2.24). Of course $M_{(25)}$ performs better than $\mathrm{M}_{\mathrm{BD}(1)}$ (since the latter only assumes a block-diagonal structure of A), but the cheaper cost of constructing and storing $\mathrm{M}_{\mathrm{BD}(1)}$ makes it a much more practical alternative. The application of the BDP to the Kuramoto Sivashinsky equation is presented in the next section.

3.5.2 The Kuramoto Sivashinsky Equation

In this section, the system with block diagonal preconditioning (3.15) is applied to a slightly modified version [45] of the Kuramoto Sivashinsky equation (KSE),

$$
\begin{aligned}
&\frac{\partial u}{\partial t} = -(u+c)\frac{\partial u}{\partial x} - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^4 u}{\partial x^4} \\
&x \in [0, L] \\
&u(0,t) = u(L,t) = 0 \\
&\left.\frac{\partial u}{\partial x}\right|_{x=0} = \left.\frac{\partial u}{\partial x}\right|_{x=L} = 0
\end{aligned}
\tag{3.18}
$$

where the length $L = 128$ to ensure chaotic solutions. The Dirichlet and Neumann boundary conditions ensure ergodicity of the system [45]. The spatial derivatives were discretised into $N+2$ nodes (N interior nodes and two boundary nodes) using second order finite difference approximations on a uniform grid (as in [45]). The resulting system was then written in the general non-linear ODE form (1.1) (see Appendix A for the derivation) and solved for $\mathbf{u}(t) = \begin{bmatrix} u_1 & u_2 & \dots & u_N \end{bmatrix}^T$. Only two grid resolutions $\delta x = 1$ ($N = 127$) and $\delta x = 0.5$ ($N = 255$) were considered. The MATLAB command 'ode45' was used for the time-stepping. Figure (3.6) shows the solution for $c = 0$ and $c = 0.8$. In both cases, organised structures with a dominant wavelength appear [64], corresponding to what is called the 'light turbulence regime'.

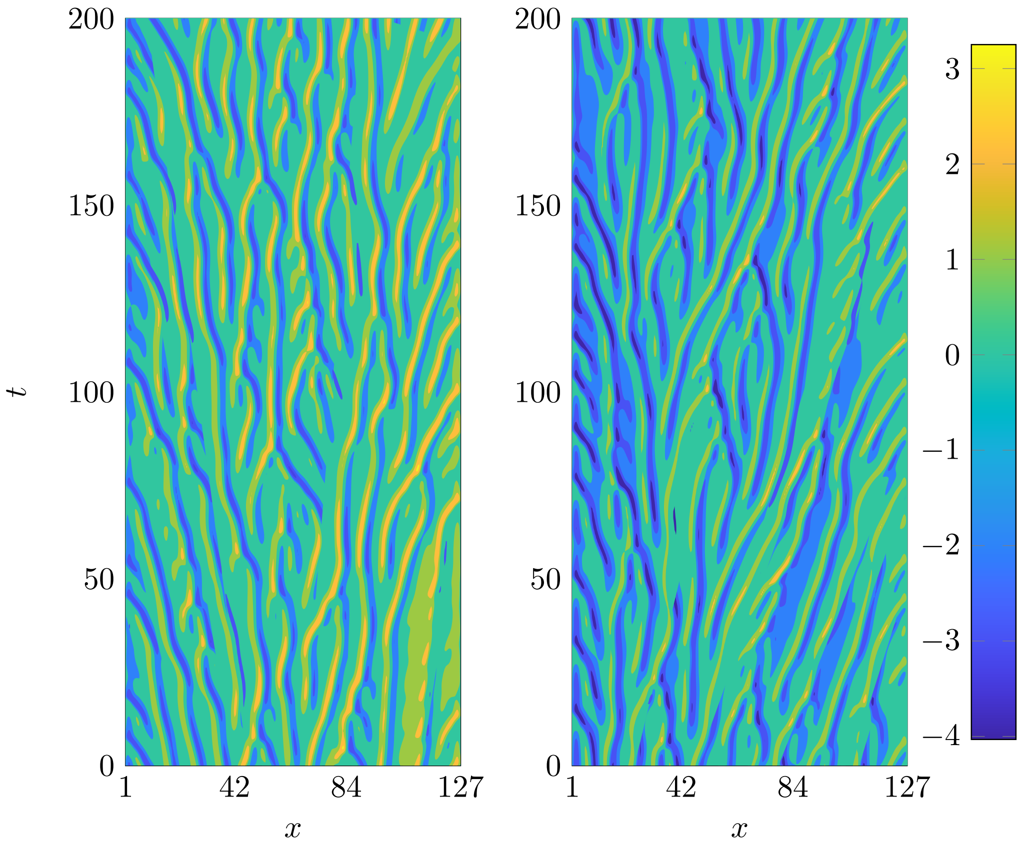


Figure 3.6: Space-time plot of the solution $u(x,t)$ for $L = 128$ using $N = 255$ nodes in the x-direction (left: $c = 0$, right: $c = 0.8$). The integration time interval is $[-1000, 200]$.

Two objectives were considered,

$$\langle \bar{u} \rangle = \frac{1}{TL}\int_0^T \int_0^L u \, dx \, dt \tag{3.19a}$$

$$\langle \overline{u^2} \rangle = \frac{1}{TL}\int_0^T \int_0^L u^2 \, dx \, dt \tag{3.19b}$$

and their sensitivities to the parameter c, $d\langle \bar{u} \rangle / dc$ and $d\langle \overline{u^2} \rangle / dc$, were sought. The MSS segment size for all cases studied was $\Delta T = 10$ (based on $\lambda_{max} \approx 0.1$ for $c = 0$ and $c = 0.8$ [45]), unless otherwise stated.

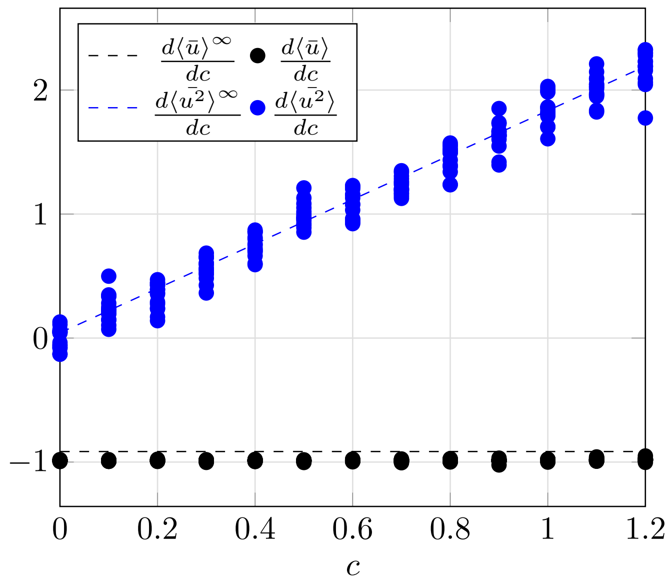


Figure 3.7: Sensitivities of $\langle \bar{u} \rangle$ and $\langle \overline{u^2} \rangle$ to the parameter c. The dashed lines were digitised from Figure (8) of [45], which were found by differentiating curve fits for $T \to \infty$. The black dots ($d\langle \bar{u} \rangle / dc$) and the blue dots ($d\langle \overline{u^2} \rangle / dc$) were obtained for $T = 100$ trajectories with random $\mathbf{u}_0$, using preconditioned MSS, for $N = 127$ and $N = 255$, respectively.

Figure (3.7) shows $d\langle \bar{u} \rangle / dc$ and $d\langle \overline{u^2} \rangle / dc$ for 20 and 15 different initial conditions $\mathbf{u}_0$, respectively. The values of the parameter c examined are between 0 and 1.2 and correspond to the 'light turbulence regime'. Each data point was computed for a randomly generated initial condition vector $0 < \mathbf{u}_0 < 1$. To obtain $\mathbf{u}(t)$, the discrete form of (3.18) was integrated in the time interval $[-1000, 100]$ and MSS was applied to $\mathbf{u}(t)$ in the time interval $[0, 100]$. The reference data [45] is the derivative of the curve fit of $\langle \bar{u} \rangle$ and $\langle \overline{u^2} \rangle$ vs. c, obtained for very long trajectories ($T = 2000$).

Figure (3.7) shows that MSS slightly under-predicts ${}^{d\langle\bar{u}\rangle}/_{dc}$ (similar to [45, 1]). This difference will be discussed in detail in the next section.

The ideal use of the BDP (3.14) involves choosing the parameters q and l such that the total number of applications of Φ_i and Φ_i^T is minimised. Considering that there are 15 positive Lyapunov exponents for $c = 0.8$ [45], it seems reasonable to choose $l = 15$ in each segment to construct (3.14). The effect of using different values of l is considered later.

Figure (3.8) shows $\sigma(A)$ and the union of the 15 computed singular values $\sigma(\Phi_i)$ for each segment i, ordered from largest to smallest. Both the converged values and the values with $q = 1$ and $q = 2$ iterations are shown. It is clear that accurate approximations to $\sigma(A)$ can be obtained with just 1 'svds' iteration (Lanczos bidiagonalisation), when $\sigma(\Phi_i)$ is evaluated. The curves start to deviate when the singular values approach unity.

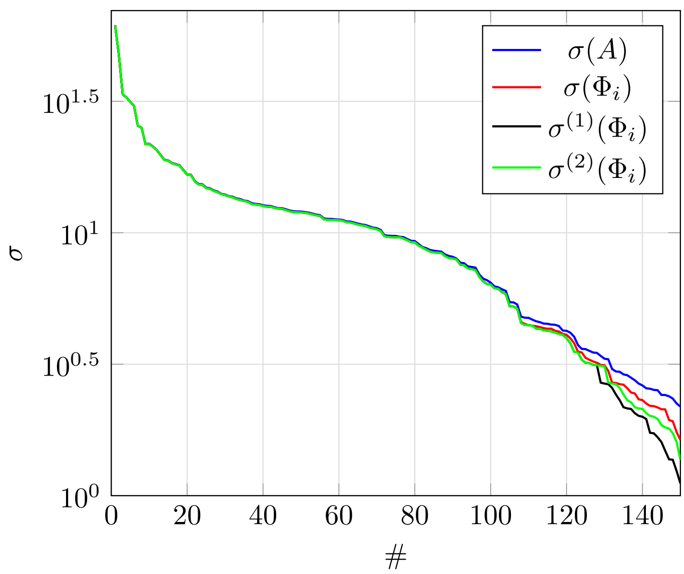


Figure 3.8: $\sigma(A)$ (blue line) and the largest 15 $\sigma(\Phi_i)$ for all segments, ordered from largest to smallest (red: exact, black: $q = 1$ iteration, green: $q = 2$ iterations). Computed for $N = 127$, $c = 0.8$ and $T = 100$ ($P = 10$).

Figure (3.9) shows the eigenvalues of the preconditioned system. Using $\mathrm{M_{BD}}{}_{(l)}^{(q)}$ has reduced the maximum eigenvalue (and therefore the condition number κ) by more than 2 orders of magnitude. The spectra of the inexact BDP systems $\mathrm{M}_{\mathrm{BD}^{(1)}_{(15)}}S$ and $\mathrm{M}_{\mathrm{BD}^{(2)}_{(15)}}S$ are very similar to the spectrum of the exact BDP system $\mathrm{M}_{\mathrm{BD}(15)}S$ (with $\mu_{max}(\mathrm{M}_{\mathrm{BD}^{(1)}_{(15)}}) \approx 15$, $\mu_{max}(\mathrm{M}_{\mathrm{BD}^{(2)}_{(15)}}S) \approx 12$ and $\mu_{max}(\mathrm{M}_{\mathrm{BD}(15)}S) \approx 6$).

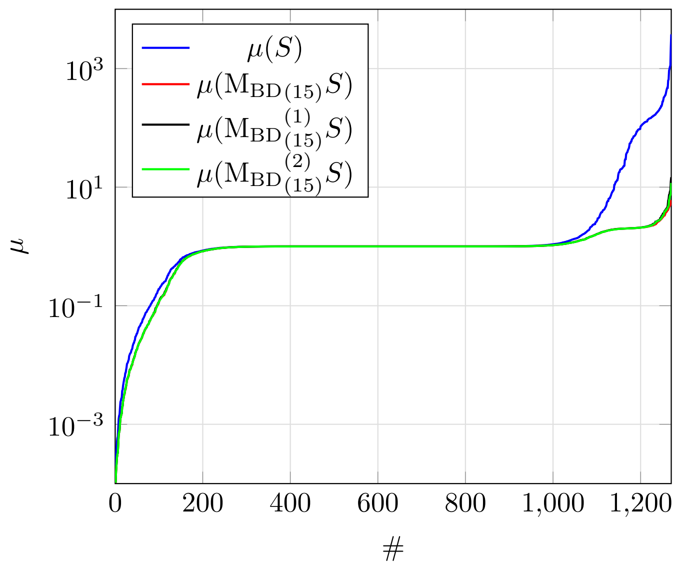
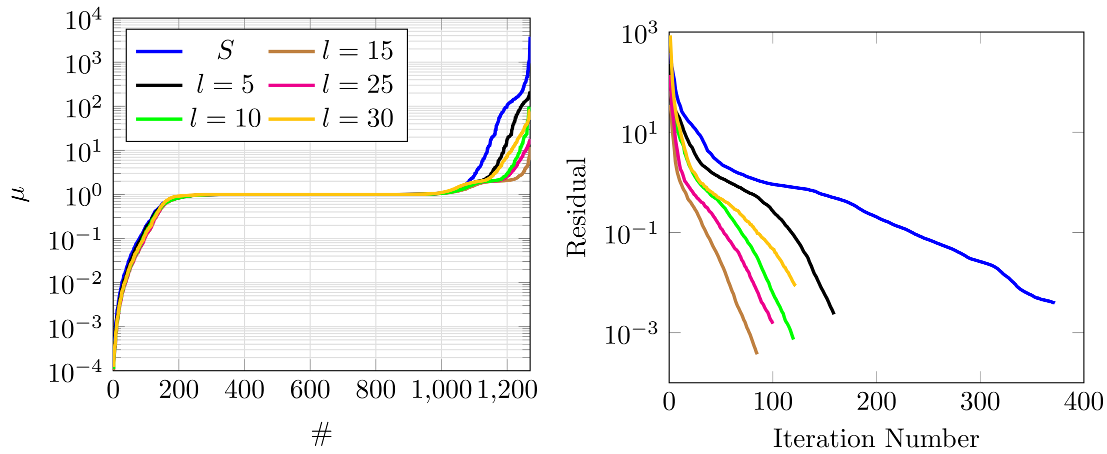


Figure 3.9: Eigenvalues of the original system (blue line), the exact BDP ($\mathrm{M}_{\mathrm{BD}(15)}$), and the inexact BDP ($\mathrm{M}_{\mathrm{BD}^{(1)}_{(15)}}$,$\mathrm{M}_{\mathrm{BD}^{(2)}_{(15)}}$) using $l = 15$.

The effect of l on the eigenvalue spectrum of $\mathrm{M}_{\mathrm{BD}^{(q)}_{(l)}}S$ and on the convergence rate



(a) Eigenvalues

(b) Residuals

Figure 3.10: Eigenvalues and residuals of the original system S (blue line) and the BDP system ($\mathrm{M}_{\mathrm{BD}^{(2)}_{(l)}}S$), for $N = 127$, $T = 100$ and $c = 0.8$. The preconditioners were constructed for different l, and their residuals were found with a regularisation value $\gamma = 0.01$ (to be introduced in Section 3.7).

is studied next. The value $q = 2$ was kept constant. Results for different l are shown in Figure (3.10). Increasing l up to $l = 15$ improves the clustering of eigenvalues and leads to faster convergence rates (Panel b). Interestingly, a further increase to $l = 25$ or $l = 30$ increases the condition number and slows down the convergence (Figure 3.10b). This indicates that after a certain value of l, adding more singular modes starts to provide unreliable information to the preconditioner $\mathrm{M_{BD}}^{(q)}_{(l)}$. This is because as l increases, the singular values approach unity and $\sigma(\Phi_i)$ starts to diverge from $\sigma(A)$. This was also observed clearly in Figure (3.1) for the Lorenz system.

3.6 Effect of the System Condition on the Accuracy of the Computed Sensitivity

In Section (3.4), a block diagonal preconditioner was presented that can deflate large singular values. While deflating is essential for accelerating the convergence, as shown in Figures (3.5) and (3.10), very small eigenvalues ($0 < \mu(S) \ll 1$) are present for quasi-hyperbolic systems [1], and cause significant issues in the solution accuracy and convergence. The presence of very small eigenvalues can be explained by reference to the homoclinic tangencies that exist for quasi-hyperbolic systems [31]. As a trajectory $\mathbf{u}(t)$ is traced in phase space, it passes through points with homoclinic tangencies (as explained in Section 1.4). If a checkpoint i lies close to a homoclinic tangency, some Lyapunov vectors become linearly dependent (i.e. some Lyapunov vectors become almost parallel to one another). As a result, the matrices Φ_i (and therefore the matrix S) become rank deficient, explaining the small eigenvalues $\mu(S)$ present for quasi-hyperbolic systems. The longer the trajectory, or the larger the number of segments, the more likely the trajectory is to pass through points with homoclinic tangencies, and the smaller $\mu_{min}(S)$ becomes. This behaviour is typical of quasi-hyperbolic systems, whereas for hyperbolic systems,

$\mu_{min}(S)$ remains bounded, as shown in [1] (because hyperbolic systems do not have homoclinic tangencies).

Consider again the sensitivity ${}^{d\langle\bar{u}\rangle}/_{dc}$ shown in Figure (3.7) for the KSE. There is a constant bias of approximately 8% for all simulations. This bias has been observed previously for the KSE in [45, 1]. Helpful insight can be gained by considering the analytical solution to the minimisation problem (2.21), expressed in terms of the singular values and left and right singular vectors of the matrix A:

$$\underline{\mathbf{v}}_l = \sum_{i=1}^{l} \left(\frac{u_i^T \underline{\mathbf{b}}}{\sigma_i} \right) v_i \tag{3.20}$$

This expression indicates that the minimal norm solution is a linear combination of the right singular vectors v_i. The coefficients ${}^{u_i^T \underline{\mathbf{b}}}/_{\sigma_i}$ are obtained by projecting the right-hand-side $\underline{\mathbf{b}}$ to the left singular vectors u_i^T and dividing by the singular value σ_i. For $l = NP$, i.e. using all singular modes, (3.20) yields the same solution obtained by solving (2.23) iteratively until convergence. However, terminating the summation at a value of $l < NP$ makes it possible to study the effect of a smaller group of singular modes. Furthermore, it is also possible to use a single value of index i to compute the contribution of an individual singular mode to the solution $\underline{\mathbf{v}}$, and therefore to the sensitivity. These properties make the decomposition (3.20) a very useful tool.

Figure (3.11) shows ${}^{d\langle\bar{u}\rangle}/_{dc}$ for the KSE, obtained when different values of l are used to terminate the summation (3.20). On the same plot, the corresponding singular values are superimposed (right vertical axis). A very interesting behaviour can be noticed. Summing modes with $\sigma \geq 1$ leads to an error in ${}^{d\langle\bar{u}\rangle}/_{dc}$ of less than 1% of ${}^{d\langle\bar{u}\rangle^\infty}/_{dc}$. When l is between 250 to 1000, with $\sigma(A) \approx 1$, the sensitivity remains almost constant. However, including in the summation terms with very small values of $\sigma(A)$ degrades the accuracy of the solution, as seen from the divergence of the squares from the dashed line. The final value is equal to the one shown in Figure

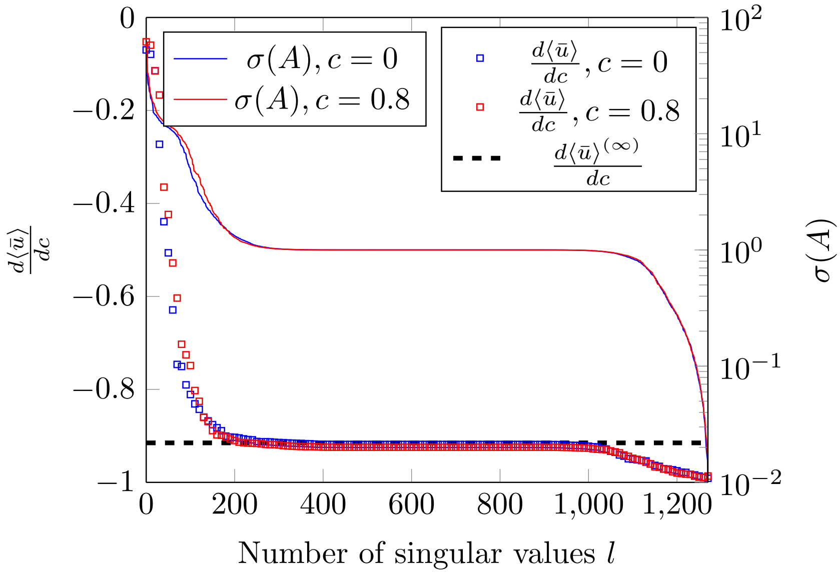
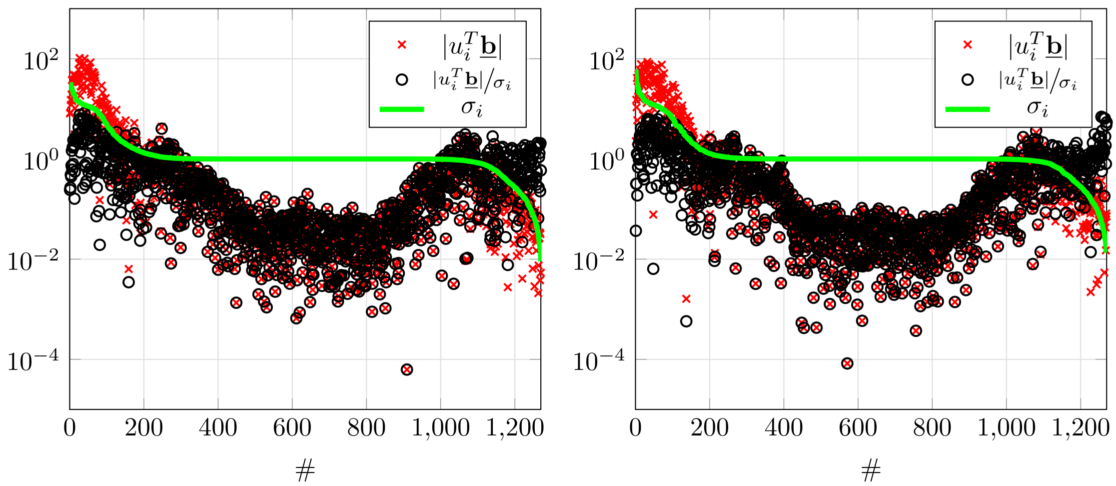


Figure 3.11: Sensitivities (left vertical axis) computed using equation (3.20) for different values of l (for the KSE: $T = 100$, $N = 127$ and $P = 10$ segments). The solid lines show $\sigma(A)$ (right vertical axis).

(3.7).



(a) $c = 0$

(b) $c = 0.8$

Figure 3.12: Spectral coefficients for $T = 100$ (KSE).

More insight can be gained by plotting the coefficients $|u_i^T \underline{\mathbf{b}}|/\sigma_i$, the projections $|u_i^T \underline{\mathbf{b}}|$ and σ_i together. This is known as a discrete Picard plot [65], and is shown in Figure (3.12). In order to interpret this plot, recall that very small singular values indicate an ill-conditioned system, i.e. that the solution is very sensitive to small changes in the right-hand-side $\underline{\mathbf{b}}$. Next, $\underline{\mathbf{b}}$ is decomposed into an unknown error-free part $\hat{\underline{\mathbf{b}}}$

and a random error part $\underline{\mathbf{e}}$ (for example due to the spatial discretisation and the time-stepping scheme), i.e. $\underline{\mathbf{b}} = \hat{\underline{\mathbf{b}}} + \underline{\mathbf{e}}$, with $\|\underline{\mathbf{e}}\|_2 \ll \|\hat{\underline{\mathbf{b}}}\|_2$. Substituting in (3.20) with $l = NP$, the following equation is obtained:

$$\underline{\mathbf{v}} = \sum_{i=1}^{NP} \left(\frac{u_i^T \hat{\underline{\mathbf{b}}}}{\sigma_i} \right) v_i + \left(\frac{u_i^T \underline{\mathbf{e}}}{\sigma_i} \right) v_i = \hat{\underline{\mathbf{v}}} + \underline{\mathbf{v}}_e \tag{3.21}$$

This equation indicates that very small σ_i can amplify the error component of the solution, $\underline{\mathbf{v}}_e = \sum_{i=1}^{NP} \left(u_i^T \underline{\mathbf{e}}/\sigma_i \right) v_i$. In order to have a meaningful solution, the projection of the error component to the left singular vector, $u_i^T \underline{\mathbf{e}}$, should decay to 0 faster than σ_i. This is known as the Picard condition [66]. Inspection of Figure (3.12a) shows that, although the values of $u_i^T \underline{\mathbf{b}}$ are quite spread out, it is clear that they decay by 3-4 orders of magnitude for i between $1 - 600$. The largest values, of order $O(10^2)$, correspond to small i, i.e. to the largest singular values. This indicates that the largest contribution to $\underline{\mathbf{b}}$ originates from the most rapidly growing modes. For $i > 600$, the values of $u_i^T \underline{\mathbf{b}}$ remain between $10^{-2} - 10^0$, i.e. at least 2 orders of magnitude smaller than the maximum. The small values of $u_i^T \underline{\mathbf{b}}$ likely originate from the random error $\underline{\mathbf{e}}$. As long as $\sigma_i \approx 1$, their contribution is innocuous, but when σ_i is reduced to 10^{-2}, they are significantly amplified (as demonstrated from the variation of $u_i^T \underline{\mathbf{e}}/\sigma_i$) and contaminate the solution. This explains the sensitivity trend in Figure (3.7). The same mechanism is valid for $c = 0.8$ (Panel b).

The above exercise was repeated for the Lorenz system. The sensitivity $d\overline{J^{(T)}}/d\rho$ was computed for $T = 100$ and $P = 200$ using equation (3.20) and compared with finite differences. The green dots in Figure (3.13) show the values of $d\overline{J^{(T)}}/d\rho$ computed using all singular modes ($l = NP = 600$). The black dots show $d\overline{J^{(T)}}/d\rho$, computed using $l = 500$ (this value of l removes all $\sigma(A) < 1$). Each dot is the average of 10 sensitivities computed using trajectories with random initial conditions $\mathbf{u}_0$. For $l = NP = 600$, there is a deviation with respect to the finite difference values (blue

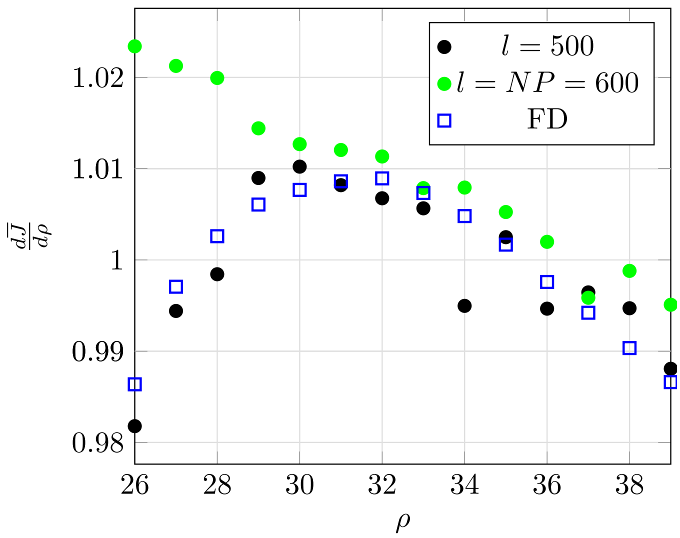


Figure 3.13: Sensitivity $d\overline{J^{(T)}}/d\rho$ against ρ for the Lorenz system. The sensitivities shown in filled dots were computed using equation (3.20) for $l = 500$ and $l = NP = 600$ (i.e. using all the singular modes). The finite difference (FD) data points (open squares) were digitised from Figure (10) of [42].

squares) of up to 3.75% (for $\rho = 26$). Note also the different trends for small values of ρ. On the other hand, the sensitivities computed using $l = 500$ follow the same trend as the finite difference data, and the errors are reduced to less than 1%.

The results presented above show, at least empirically, that the sensitivity bias suffered by MSS is related to the singular modes of the matrix A with values $\sigma(A) < 1$. Truncation of these modes using the solution form (3.20) eliminates this bias almost entirely. It is expected that this would be possible in general for quasi-hyperbolic systems with well-defined sensitivities. Although computing the partial SVD of the matrix A would be very expensive in practice, this was an important finding worth reporting and investigating further.

3.7 Regularisation of the Preconditioned System

The block diagonal preconditioning approach (3.15) has suppressed the largest singular values, but has not affected the smallest ones. In order to further improve the convergence rate, the system needs to be regularised, i.e. the very small eigenvalues need to be filtered out. Tikhonov regularisation is one of the most widely used regularisation techniques. The idea is to solve a regularised version of (2.24),

$$(\gamma I + S)\underline{\mathbf{w}} = \underline{\mathbf{b}} \tag{3.22}$$

where $\gamma > 0$ is an appropriately chosen parameter. Equation (3.22) is derived by augmenting the minimisation statement in (2.21) as follows:

$$\underset{\mathbf{v}_i}{\text{Minimise}} \ \frac{1}{2}\sum_{i=0}^{P} \|\mathbf{v}_i\|_2^2 + \gamma\|\mathbf{w}_i\|_2^2 \tag{3.23}$$

This form of regularisation has been employed in [1] to improve the conditioning of S. The physical interpretation is that the parameter γ relaxes the continuity constraint (2.21b) and shifts all $\mu(S)$ to $\mu(S) + \gamma$. A large γ value over-relaxes the constraint (2.21b), leading to inaccurately computed sensitivities. If γ is chosen adequately, it can improve the accuracy and accelerate the convergence simultaneously, both for very little additional cost.

In this section, Tikhonov regularisation is applied to the preconditioned system (3.15) to regularise small $\mu(S)$. There are two options: either to regularise the original system and then apply preconditioning, i.e. to solve

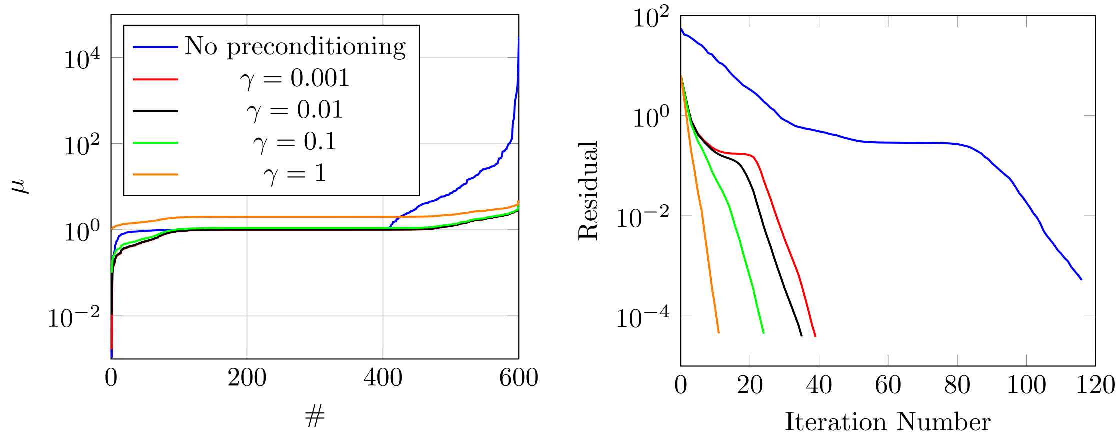
$$\mathrm{M_{BD}}_{(l)}^{(q)} \left(\gamma I + S\right) \underline{\mathbf{w}} = \mathrm{M_{BD}}_{(l)}^{(q)} \underline{\mathbf{b}} \Longrightarrow \left(\gamma \mathrm{M_{BD}}_{(l)}^{(q)} + \mathrm{M_{BD}}_{(l)}^{(q)} S\right) \underline{\mathbf{w}} = \mathrm{M_{BD}}_{(l)}^{(q)} \underline{\mathbf{b}} \tag{3.24}$$

or to apply preconditioning first and then regularise, i.e. to solve

$$\left(\gamma I + \mathrm{M_{BD}}_{(l)}^{(q)} S\right) \underline{\mathbf{w}} = \mathrm{M_{BD}}_{(l)}^{(q)} \underline{\mathbf{b}} \tag{3.25}$$

When the exact preconditioner $M_{(l)}$ (3.8) is used, and $\gamma \ll \mu_{max}\left(M_{(l)}S\right)$, the two options are almost identical. For $\mathrm{M_{BD}}_{(l)}^{(q)}$, the second option guarantees the clustering of eigenvalues inside the tight range, $\left[\gamma + \mu_{min}(\mathrm{M_{BD}}{}^{(q)}_{(l)} S), \gamma + \mu_{max}(\mathrm{M_{BD}}{}^{(q)}_{(l)} S)\right]$. Such tight clustering is conducive to rapid convergence of iterative subspace solvers. The second option physically means that the rapidly growing modes within each segment are deflated first and then the continuity constraint between segments is slightly relaxed. This leads to a smaller number of iterations than solving (3.24).

The combined effect of regularization and preconditioning for the Lorenz system can be seen in Figure (3.14a). Without regularization, the condition number $\kappa(S) \approx 3 \times 10^7$ while $\kappa(\mathrm{M_{BD(1)}} S) \approx 5.6 \times 10^3$, i.e. a reduction of 4 orders of magnitude in κ. When using $\gamma = 1$, $\kappa(I + \mathrm{M_{BD(1)}} S) \approx 4$, and convergence is obtained in 12 iterations only (Figure 3.14b).



(a) Eigenvalues

(b) Residuals

Figure 3.14: Eigenvalues and residuals of the original system S and the preconditioned system $\gamma I + \mathrm{M_{BD(1)}} S$ for different γ. A Lorenz system trajectory length $T = 200$ with $\Delta T = 1$ was used for $\rho = 40$.

The sensitivities $\overline{dJ^{(T)}}/d\rho$ computed for different γ are shown in Figure (3.15). For reference, $\overline{dJ^{(\infty)}}/d\rho$ and the solution for $\gamma = 0$ (which is affected by the ill-conditioning of S) are shown. There is a range $0.001 \leq \gamma \leq 0.1$ which provides an adequate balance between filtering out the noisy singular values, while keeping $\|\underline{\mathbf{b}} - A\underline{\mathbf{v}}\|_2$ close to zero. In this range, the solution error is $O(\leq 2\%)$. Further increase to $\gamma = 1$ relaxes the constraint $A\underline{\mathbf{v}} = \underline{\mathbf{b}}$ significantly, and results in a solution error of $O(5\%)$. A method to estimate the optimal value of γ based on the L-curve criterion was proposed in [67, 68].

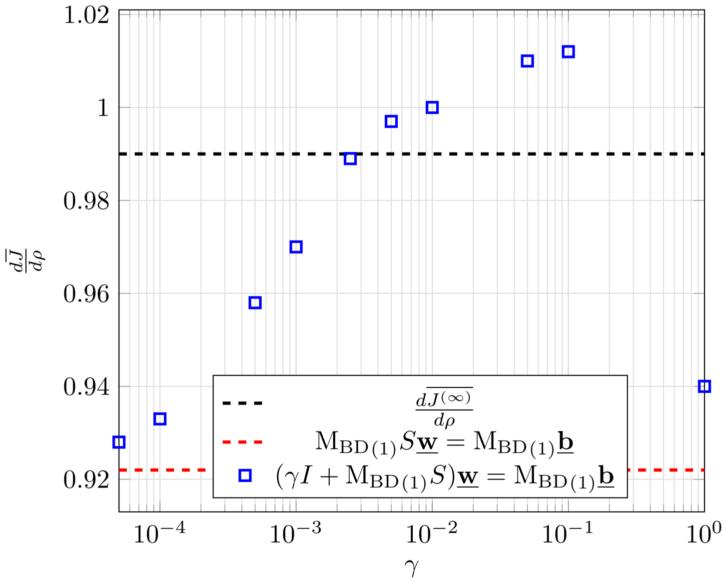


Figure 3.15: Sensitivities for the Lorenz system ($T = 200$, $\Delta T = 1$ and $\rho = 40$) for different γ. The value of $d\overline{J^{(\infty)}}/d\rho$ (black-dashed line) was digitised from Figure (5d) of [22].

Ideally, the convergence rate should be independent of T (and therefore the number of segments P) and the number of degrees of freedom N. This would make MSS applicable to large systems. Figure (3.16) shows that indeed, the convergence rate is almost independent of T for the Lorenz system. Theoretical analysis shows that for a linear system with a symmetric, positive definite matrix $\mathscr{A}$, the $\mathscr{A}$-norm of the error at iteration m, $\|r_m\|_{\mathscr{A}}$, satisfies (refer to [50]):

$$\frac{\|r_m\|_{\mathscr{A}}}{\|r_0\|_{\mathscr{A}}} \leq 2 \left(\frac{\sqrt{\kappa\left(\mathscr{A}\right)} - 1}{\sqrt{\kappa\left(\mathscr{A}\right)} + 1} \right)^m \tag{3.26}$$

This error bound is independent of the number of unknowns, and provided that $\kappa\left(\mathscr{A}\right)$ is suppressed by preconditioning and regularisation, the number of iterations becomes independent of T and N. This is clearly demonstrated in Figure (3.16). Tests showed that using equation (3.26) to predict the number of iterations that will result in a pre-specified relative residual (set to 10^{-5}), overestimated the actual iterations needed in practise. This shows that (3.26) indeed provides a very conservative upper bound. The bound becomes more realistic as $\kappa(\mathscr{A})$ decreases. Table (3.1) shows that for the chosen $\gamma = 0.1$, $\overline{dJ^{(T)}}/d\rho \to 0.99$ (the infinite time-averaged sensitivity, $\overline{dJ^{(\infty)}}/d\rho$).

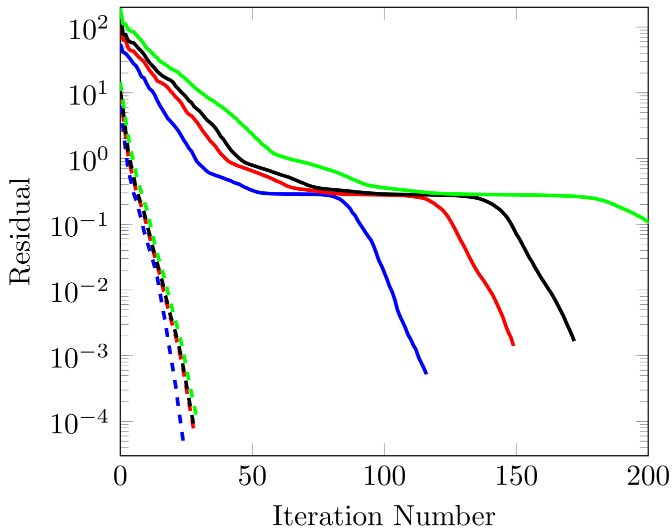
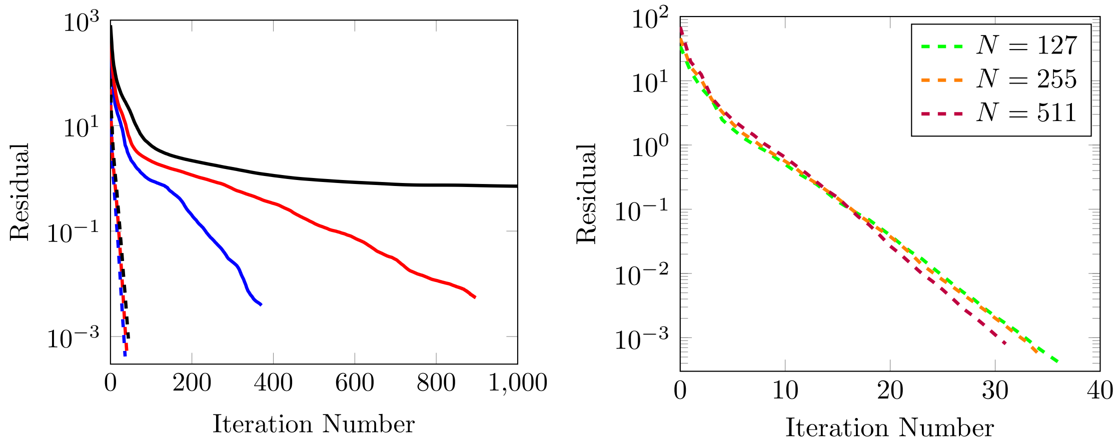


Figure 3.16: Residuals for S (solid lines) and the BDP system $\gamma I + \mathrm{M}_{\mathrm{BD}(1)}S$ (dashed lines), with $\gamma = 0.1$. The segment size is $\Delta T = 1$ and $\rho = 40$ (Lorenz system). Blue: $T = 200$, red: $T = 300$, black: $T = 500$, green: $T = 1000$.

	$T = 200$	$T = 300$	$T = 500$	$T = 1000$
$\frac{d\overline{J^{(T)}}}{ds}$	1.01	0.97	0.97	0.99

Table 3.1: A table showing $\overline{dJ^{(T)}}/d\rho$ for $\rho = 40$ (Lorenz system), using different trajectory lengths. A regularisation value $\gamma = 0.1$ was used.

Figure (3.17) shows the convergence rates for varying T (Panel a) and N (Panel b) for the KSE. The figure demonstrates again that the combination of regularisation and preconditioning (dashed lines) renders the convergence almost independent of T (with $N = 127$) and N (with $T = 100$). The fast convergence is a direct result of the clustering of eigenvalues in a tight range, and hence the suppression of the condition number $\kappa(S)$.



(a) Varying T with $N = 127$ (b) Varying N with $T = 100$

Figure 3.17: Residuals of the original system (solid lines), and of the system $\gamma I + \mathrm{M_{BD}}_{(l)}^{(q)} S$, with $\gamma = 0.09$, $q = 2$ and $l = 15$ (dashed lines). Blue: $T = 100$, red: $T = 200$, black: $T = 500$. The GMRES solver used a relative tolerance of 1×10^{-5}.

3.8 Computational Cost

In this section, the computational cost of preconditioned MSS is considered. Only the costs of constructing the BDP (3.14) and of solving the Schur complement (3.25) are considered, since these operations account for most of the wall time. The total number of applications of Φ_i and Φ_i^T (per segment), is used to quantify the cost. It is assumed that these operations can be done in parallel using P processors (one processor allocated to each segment). It is also assumed that message passing can be overlapped with computation and that it is completed before the end of the computation.

The cost (number of applications of Φ_i and Φ_i^T) of constructing one block of the preconditioner, $\mathrm{M}_{(l),i}^{(q)}$, is $2q(l+2)$, where $l+2$ denotes the selected size of the subspace. The factor 2 appears because the partial singular value decomposition of Φ_i requires matrix-vector products with both Φ_i and Φ_i^T. One iteration of

for the solution of (3.25) requires the application of A and A^T once, i.e., the solution to (3.25) requires m applications of Φ_i and Φ_i^T per segment, where m is the number of iterations to convergence. This cost is only approximate, as the time taken to orthogonalise the subspace every time a new vector is added has been neglected.

Table (3.2) shows a cost comparison for different T (for the KSE), with and without preconditioning (the values correspond to Figure 3.17a). The parameters $q = 2$ and $l = 15$ were used, so the preconditioner cost per segment is $2q(l + 2) = 68$ total Φ_i and Φ_i^T applications. It can be seen that the combination of preconditioning and regularisation results in significant savings; the cost is reduced by a factor of 35 for $T = 500$. Note that the condition number is reduced by between 5 to 7 orders of magnitude (depending on T) and remains almost constant. The number of iterations depends very weakly on T, and so does the total cost per segment.

	$T = 100$ ($P = 10$)	$T = 200$ ($P = 20$)	$T = 500$ ($P = 50$)
Cost of solving $\left(\gamma I + \mathrm{M_{BD}}{}_{(l)}^{(q)} S\right) \underline{\mathbf{w}} = \mathrm{M_{BD}}{}_{(l)}^{(q)} \underline{\mathbf{b}}$			
μ_{max}, μ_{min}	11.87, 0.090	11.87, 0.090	13.37, 0.090
Condition number, κ	132	132	149
Number of iterations, m	38	42	46
Cost of iterations	76	84	92
Cost of constructing $\mathbf{M_{BD}}{}_{(l)}^{(q)}$	68	68	68
Total cost	144	152	160
Cost of solving $S\underline{\mathbf{w}} = \underline{\mathbf{b}}$			
μ_{max}, μ_{min}	3800, 1.90×10^{-4}	3800, 1.80×10^{-5}	4900, 4.90×10^{-6}
Condition number, κ	2.0×10^7	2.1×10^8	1.0×10^9
Number of iterations, m	371	897	2,790
Total cost	742	1,794	5,580

Table 3.2: A table showing the cost (total number of Φ_i and Φ_i^T applications per segment) for the cases shown in Figure (3.17a). The preconditioner was constructed using $q = 2$, $l = 15$ and $\gamma = 0.09$. The relative residual $\|r_m\|_2/\|r_0\|_2 \approx 1 \times 10^{-5}$ for all cases.

The minimum and maximum eigenvalues are also reported in Table (3.2), and this information can be used to assess the individual effects of preconditioning and regularisation. Preconditioning results in a reduction of μ_{max} by two orders of magnitude (and a corresponding reduction in κ). Regularisation raises μ_{min} by three to five orders of magnitude.

There is further scope for cost reduction. For example, for $T = 500$, a value $\gamma = 0.09$ was chosen, which produces a value for the sensitivity ${}^{d\langle\bar{u}\rangle^\infty}/_{dc}$ accurate to 1%. Increasing to $\gamma = 0.25$ and using $q = 1$ still gives an acceptable sensitivity (the error is 8%), but the number of iterations is reduced to 35, and the total cost is 116 Φ_i and Φ_i^T applications per segment (down from 160).

It is important to try to predict the cost of the preconditioned MSS algorithm for higher dimensional turbulent systems. Tests showed that the optimal value of preconditioning modes l that minimised the total approximate cost (the solution to (3.25) and constructing the preconditioner (3.14)), was close to N_{+LE}. Figure (3.18) shows that the optimal value to use was $l = 18$ ($N_{+LE} = 15$), while further increase in l led to a rise in cost. It is expected (in general), that if $l = N_{+LE}$ and a suitable value for γ is chosen, the eigenvalues would always remain clustered within a narrow range, and the cost of solving (3.25) would be independent of T or N. However, the cost of computing the preconditioner will increase linearly with N_{+LE}, and would likely dominate the total cost for larger systems, since all modes would need to be computed to minimise the total cost.

It has been shown that N_{+LE} varies with the mesh size [35, 69] and Reynolds number [70]. Ni [35] showed that for a low Reynolds number flow around a cylinder, the Lyapunov spectrum was similar for two mesh sizes, but the finer mesh had a smaller N_{+LE}. In [69], where the authors computed the Lyapunov spectrum for a separated flow over an airfoil, N_{+LE} was shown to increase with N_{+LE}, contrary to the findings in [35]. It is not surprising that N_{+LE} will increase with the Reynolds number, as shown for example in [70].

Based on the observations outlined above, applying preconditioned MSS to complex turbulent systems with thousands of positive exponents may still be very difficult due to the large expected cost of preconditioning with $l = N_{+LE}$. However, this needs further investigation.

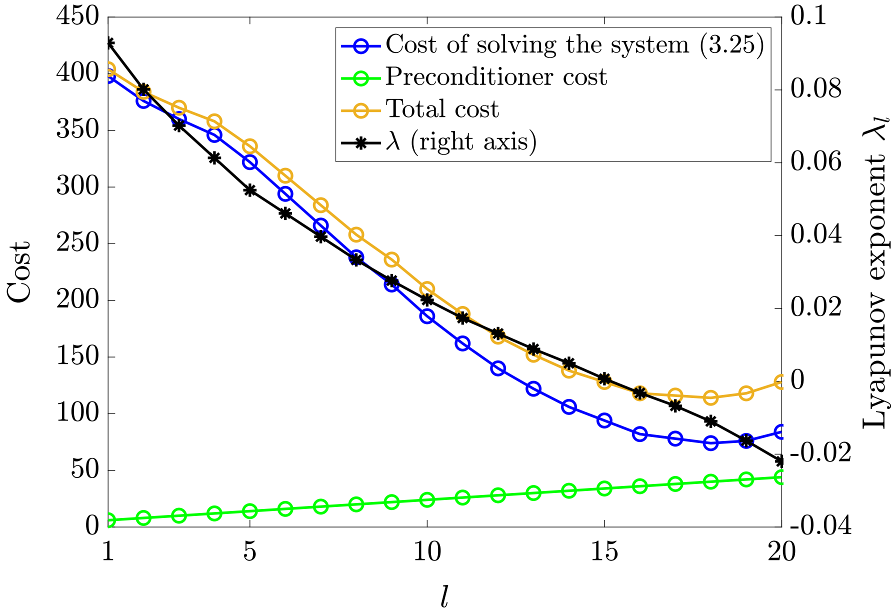


Figure 3.18: The cost (number of applications of Φ_i and Φ_i^T per segment) for different preconditioning modes l. The trajectory was computed for $T = 200$ and $c = 0.8$. The solver and preconditioner used $\gamma = 0.09$, $\Delta T = 10$ and $q = 1$. The Lyapunov exponents (right axis) have been digitised from Figure (9) of [45].

3.9 Summary

In this chapter, a block diagonal preconditioner was derived to accelerate the convergence rate for the solution of the linear system (2.24) arising from the application of the MSS algorithm. The preconditioner is based on the partial singular value decomposition of the diagonal blocks (3.11) of the constraint matrix A (2.22). The preconditioner was applied to the Lorenz system and the KSE.

The number of singular modes to retain in the partial SVD, l, is case dependent, and a well-chosen value is required for fast convergence. If the number of positive Lyapunov exponents is known, it can be used to inform the choice of l. Strictly speaking, however, this is not necessary, and l can be chosen based on the number of $\sigma(\Phi_i) > 1$.

When the preconditioner was combined with a regularisation method, the condition number was significantly suppressed, and the convergence rate was found to be weakly dependent on the number of degrees of freedom and the length of the trajectory. The total number of operations was significantly reduced as a result. This weak dependence on N and T paves the way to apply MSS to higher dimensional turbulent systems for sensitivity analysis and optimal control applications. Furthermore, it was shown that a large condition number can affect the accuracy of the computed sensitivity, and can explain an 8% sensitivity bias for the KSE in the light turbulence regime. It can also explain a smaller bias for the Lorenz system.

Chapter 4

Feedback Control of Chaotic Systems using Shadowing[1]

4.1 Introduction

Computing feedback controllers for large chaotic systems is very challenging. Standard Linear Quadratic Regulator (LQR) theory is often inapplicable, or prohibitively expensive because the computational cost scales as $O(N^3)$. In this chapter, an algorithm based on shadowing is proposed for computing feedback controllers for chaotic systems and is applied to the KSE. The algorithm relies on the sensitivities computed by preconditioned MSS to find the optimum feedback control matrix elements iteratively.

This chapter is structured as follows: in Section (4.2), the challenges of applying Linear Quadratic Regulator (LQR) theory to nonlinear systems are discussed. LQR theory is summarised, and the proposed control algorithm is introduced in Section

[1]Results from this chapter have appeared in [71]: K. Shawki and G. Papadakis. Feedback control of chaotic systems using multiple shooting shadowing and application to Kuramoto–Sivashinsky equation. Proceedings of the Royal Society A, 476(2240), 2020. http://dx.doi.org/10.1098/rspa.2020.0322

(4.3). In Section (4.4), the feedback control kernels computed using both methods, as well as the system response to both controllers are compared and discussed. The algorithm performance is discussed in Section (4.5).

4.2 Feedback Control of Nonlinear Systems

Actuation of linear or nonlinear systems to meet a desired objective has many applications, including transition delay [72], control of separation [73] and drag reduction [7]. For linear systems, optimal control theory (for example, LQR or LQG) is very well developed. A quadratic objective function that includes the cost of actuation is defined and minimised subject to the linear constraints. The optimal feedback matrix K is obtained from the solution to an Algebraic Riccati Equation (ARE). For nonlinear systems, a linearisation process around the target state is first carried out, and then the linear optimal control theory can be applied. Once a controller has been derived, it can be applied to the full nonlinear system. When all the system states are available, and there are no uncertainties, the linear theory is known as Linear Quadratic Regulator (LQR) theory [74]. This theory can be extended to handle unknown system dynamics (modelled as white, Gaussian noise) and can be coupled with an estimator (also based on a quadratic objective), that can extract the state of the system from noisy measurements. This is known as the Linear Quadratic Gaussian (or LQG) approach [74].

LQR and LQG are effective for turbulent systems with energy conserving nonlinearities, such as channel flows [75, 76, 77, 78, 79]. In such flows, the nonlinear term, which is responsible for the transfer of energy from large to small scales, vanishes when integrated in the whole domain. Usually, the aim of LQR (or LQG) is to minimise the kinetic energy of the fluctuations around a target state, given a distribution of sensors and actuators mounted on the walls. LQG has also been used for the suppression of 2D disturbances [80] and Tollmien Schlichting waves [72] on flat plate geometries and for the control of separation in a square cavity [73]. Applying linear control theory assumes that the linearised Navier-Stokes equations capture the important dynamical processes [81]. Linear models, however, cannot account for the cascade of energy from large to small scales, which is of course a nonlinear phenomenon.

An important issue associated with LQR and LQG is the computational cost. The feedback matrix K and the Kalman gain L (associated with estimator of LQG) both require a solution to an ARE which has computational complexity $O(N^3)$. Its solution becomes intractable for $N > 10^3$ [81]. Reduced Order Models (ROM) have been proposed to reduce the cost. The idea is to construct lower-order approximations to the plant's input-output dynamics (see for example [73, 72, 76]). Modes deemed unnecessary are truncated, thus reducing the system dimension. In [82, 83, 84], feedback control of the KSE using a truncated Galerkin's method for spatial discretisation was used. Truncation was based on the number of unstable modes, which were separated from the stable ones. The number of control actuators was chosen based on the number of unstable modes. The issue with ROMs, however, is that there is no guarantee that the truncated modes will not affect the objective function [81]. Furthermore, ROMs only capture the dynamics of the uncontrolled flow (open loop) and not that of the controlled flow (closed loop) [85], which may be quite different.

A few attempts have been made to bypass the solution of the Riccati equation and therefore the need for a ROM. These approaches compute directly the elements of the feedback matrix K using iterative methods that rely on the integration of the linearised governing and adjoint equations in a forward/backward loop. They scale to large N and are therefore suitable for systems arising from the discretisation of the Navier-Stokes equations in complex domains. In [86, 87], the authors proposed a method that can compute the elements of the matrix row-by-row. It is based on the iterative computation of the adjoint of a forward problem, the latter defined for the direct-adjoint vector pair associated with the LQR problem. The method was extended to the estimation problem in [85] and applied successfully to a 2D boundary layer, while in [88], the method was extended to robust (H_∞) control. In [89], an adjoint method for simultaneously updating all the feedback matrix elements was proposed. The cost function of the LQR problem was written in terms of K, and the latter was computed iteratively using forward/backward marching of the governing and adjoint equations.

All the above approaches have been applied to linear, time-invariant system constraints. In this chapter, the nonlinear governing equations (1.1) are used as constraints to compute K iteratively. The matrix is extracted from the system attractor and no model reduction is applied (an area which is far less well developed for nonlinear systems compared to linear ones). The application of the iterative approach to extract K is not straightforward for chaotic systems, however. The reason for this is the backward-in-time exponential growth of the adjoint variables, as discussed in Section (1.5).

In this chapter, the Preconditioned Multiple Shooting Shadowing (PMSS) method is used to compute the adjoint variables and from them, in a single computation, all the elements of the optimal feedback control matrix. The idea is applied to the KSE with Dirichlet boundary conditions, that result in an ergodic system [45]. There is also an important additional benefit. For this type of boundary conditions, the non-linearity of the system is energy conserving, a property which makes standard methods, like LQR, effective. Thus it is possible to compare the control kernels produced by the shadowing and LQR approaches. It should be noted that very few attempts have been made to solve an optimisation problem using LSS before [23, 90], and none has considered the computation of a feedback matrix.

4.3 Formulation of the Control Problem

For control purposes, an actuation $\mathbf{s}(t)$ which will modify the behaviour of the system (1.1) to meet a desired objective is sought. Assuming M actuators (in which case $\mathbf{s}(t)$ becomes a vector of length M), the controlled system takes the form

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}) + B\mathbf{s}(t) \tag{4.1}$$

where B is the input matrix (size $N \times M$) that determines the spatial distribution of the actuation. Moreover, it is assumed that the actuation $\mathbf{s}(t)$ takes the linear feedback control form

$$\mathbf{s}(t) = -K\left(\mathbf{u}(t) - \mathbf{u}_{targ}\right) \tag{4.2}$$

where K is an $M \times N$ feedback matrix and $\mathbf{u}_{targ}$ is the desired (target) state of the system. The controlled system then becomes

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}) - BK\left(\mathbf{u}(t) - \mathbf{u}_{targ}\right) \equiv \mathbf{h}(\mathbf{u}, K) \tag{4.3}$$

where for future reference, $\mathbf{h}(\mathbf{u}, K) = \mathbf{f}(\mathbf{u}) - BK\left(\mathbf{u}(t) - \mathbf{u}_{targ}\right)$. The objective is to compare the matrix K obtained using standard linear optimal control theory (LQR) with the one obtained from Preconditioned Multiple Shooting Shadowing (PMSS). Both approaches are described below.

4.3.1 Control using Linear Quadratic Regulator (LQR)

Linear Quadratic Regulator (LQR) theory is developed for linear time-invariant (LTI) systems. The governing ODE set (1.1) is first linearised around a stationary target state, $\mathbf{u}_{targ}$, i.e. $\mathbf{u} = \mathbf{u}_{targ} + \mathbf{u}'$ is substituted into (1.1), and keeping only the linear terms, one arrives at

$$\frac{d\mathbf{u}'}{dt} = A_l \mathbf{u}' \tag{4.4}$$

where $A_l = {}^{\partial \mathbf{f}}/_{\partial \mathbf{u}}|_{u_{targ}}$ is the Jacobian matrix evaluated at the target sate. Actuation is then applied to the linearised system, i.e.,

$$\frac{d\mathbf{u}'}{dt} = A_l \mathbf{u}' + B\mathbf{s}(t) \tag{4.5}$$

LQR finds the optimum actuation $\mathbf{s}(t)$ that minimises the following objective:

$$J^{(\infty)} = \int_0^\infty \left(\mathbf{u}'^T Q \mathbf{u}' + \mathbf{s}^T R \mathbf{s} \right) \, dt \tag{4.6}$$

subject to (4.5), where Q and R are weighting matrices (size $N \times N$ and $M \times M$, respectively). Both are symmetric and $Q \geq 0$ and $R > 0$ (i.e. positive semi-definite and definite respectively). The solution to the optimisation problem results in an optimum actuation $\mathbf{s}(t)$ that is related to the state $\mathbf{u}(t)$ via equation (4.2). The feedback matrix K is obtained from

$$K = R^{-1} B^T P \tag{4.7}$$

where the matrix P (size $N \times N$) is the solution to the algebraic Riccati equation,

$$A_l^T P + P A_l - P B R^{-1} B^T P + Q = 0 \tag{4.8}$$

The cost of the solution of this equation scales as $O(N^3)$. In the present work, $B = I$ and the weighting matrices $Q = I$ and $R = cI$, where c is a positive constant. The optimal feedback controller is then given by

$$K = \frac{1}{c} P \tag{4.9}$$

The derivation of equations (4.7) and (4.8) can be found in standard control textbooks [91, 74]. The optimum feedback controller (4.7) is then applied to the full nonlinear system (4.3). For LTI systems, the linear relationship (4.2) between the optimum actuation and state is exact, and is a direct outcome of the solution of the optimisation problem. In the present work, the target state is set to $\mathbf{u}_{targ} = \mathbf{0}$, i.e. the actuation aims to bring the system to rest.

4.3.2 Control using Adjoint Preconditioned Multiple Shooting Shadowing (PMSS)

In this section, a framework for finding the control matrix K that solves the nonlinear control problem:

$$\begin{aligned} \underset{K}{\text{Minimise}} \quad & \overline{J^{(\infty)}}(\mathbf{u}, K) = \lim_{T\to\infty} \frac{1}{T}\int_0^T J(\mathbf{u}, K)\, dt && (4.10a) \\ \text{subject to} \quad & \frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}) - BK\mathbf{u} \left(= \mathbf{h}(\mathbf{u}, K)\right) && (4.10b) \end{aligned}$$

is introduced. In analogy with (4.6), $J(\mathbf{u}, K)$ is defined as

$$J(\mathbf{u}, K) = \mathbf{u}^T Q\mathbf{u} + \mathbf{s}^T R\mathbf{s} = \mathbf{u}^T Q\mathbf{u} + (K\mathbf{u})^T R\,(K\mathbf{u}) \tag{4.11}$$

Note that in (4.11), the full state $\mathbf{u}$ appears, not the perturbation around the target $\mathbf{u}'$, as in (4.4). Since however $\mathbf{u}_{targ} = \mathbf{0}$ has been chosen, both control problems are aiming to bring the system to rest.

The optimisation problem (4.10) can be solved iteratively using the sensitivities $d\overline{J^{(\infty)}}/dK$ and coupling with an updating method, for example gradient descent,

$$K^{(i+1)} = K^{(i)} - a^{(i)} \left(\frac{d\overline{J^{(\infty)}}}{dK}\right)^{(i)} \tag{4.12}$$

where i is the iteration number and $a^{(i)}$ is a step size. The iterative algorithm is summarised below.

Algorithm 3: PMSS Control

Inputs: T, $\mathbf{u}_0$, $K^{(0)}$, ϵ. Output: K (converged control matrix)

1. Set $i = 0$.
2. Integrate $\dot{\mathbf{u}} - \mathbf{h}(\mathbf{u}, K^{(0)}) = 0$ with initial condition $\mathbf{u}(0) = \mathbf{u}_0$ in the interval $[0, T]$ to obtain $\mathbf{u}^{(0)}(t)$. Compute $\overline{J^{(T)}}^{(0)}$.
3. Call **Adjoint PMSS solver** to obtain $\left(d\overline{J^{(T)}}/dK\right)^{(i)}$ for the trajectory $\mathbf{u}^{(i)}(t)$.
4. Compute $a^{(i)}$ using an appropriate algorithm (for example backtracking or Brent's method [92]).
5. Update the control parameters: $K^{(i+1)} = K^{(i)} - a^{(i)} \left(d\overline{J^{(T)}}/dK\right)^{(i)}$.
6. Integrate $\dot{\mathbf{u}} - \mathbf{h}(\mathbf{u}, K^{(i+1)}) = 0$ and store $\mathbf{u}^{(i+1)}(t)$. Compute $\overline{J^{(T)}}^{(i+1)}$ (Note: This step is included within Step 4 if backtracking is employed).
7. If $\left|\frac{\overline{J^{(T)}}^{(i+1)} - \overline{J^{(T)}}^{(i)}}{\overline{J^{(T)}}^{(i)}}\right| < \epsilon$ break, and output $K = K^{(i+1)}$. If not, set $i = i + 1$ and return to Step 3.

The sensitivity matrix $d\overline{J^{(\infty)}}/dK$ is computed using adjoint MSS (see Section 2.4.2). In this way, $d\overline{J^{(\infty)}}/dK$ is found with a single adjoint solution. The block diagonal preconditioner (3.14) is used to precondition the adjoint MSS Schur complement (2.31) as follows:

$$\left(\gamma I + \mathrm{M_{BD}}{}_{(l)}^{(q)} S\right) \underline{\hat{\mathbf{w}}} = -\mathrm{M_{BD}}{}_{(l)}^{(q)} (A\underline{\mathbf{g}}) \tag{4.13}$$

Only the right-hand-side of (4.13) differs from the preconditioned tangent Schur complement (3.25). Appendix (B) shows how to compute the MATVEC product $\left(\gamma I + \mathrm{M_{BD}}{}_{(l)}^{(q)} S\right) \underline{\mathbf{z}}$ for an arbitrary vector $\underline{\mathbf{z}}$. The steps required to compute $d\overline{J^{(\infty)}}/dK$ are summarised in Algorithm (4).

Step 4 of Algorithm (3) typically requires multiple evaluations of (4.10b) to compute $a^{(i)}$. For this study, backtracking is used, which guarantees the monotonic reduction

Algorithm 4: Adjoint PMSS Solver

Inputs: T, P, l, q, γ. Output: $\frac{d\overline{J^{(T)}}}{dK}$

1. Compute the vector $A\underline{\mathbf{g}}$ (Refer to Appendix (B) for the computation details).
2. Use (3.14) to compute the preconditioner blocks $\mathrm{M}_{(l),i}^{(q)}$.
3. Form the RHS of (4.13), $-\mathrm{M_{BD}}_{(l)}^{(q)}(A\underline{\mathbf{g}})$.
4. Solve (4.13) for $\underline{\hat{\mathbf{w}}}$ using GMRES.
5. Integrate (2.32) backwards in time (for all segments $i = 1 : P$) with the terminal conditions in (2.33) to find $\hat{\mathbf{w}}(t)$.
6. Compute $d\overline{J^{(T)}}/dK$ by evaluating (2.34) NM times (i.e. once for each element of K).

* Note that $\mathbf{f}(\mathbf{u})$ must be replaced by $\mathbf{h}(\mathbf{u}, K) = \mathbf{f}(\mathbf{u}) - BK\mathbf{u}$ in the equations of Section (2.4.2) referred to above.

of $\overline{J^{(T)}}^{(i)}$, and is considerably less expensive than other extrema finding methods (such as Brent's method [92]). Backtracking requires the computation of $\mathbf{u}(t)$ and $\overline{J^{(T)}}$, so these are obtained on Step 4 (and make Step 6 redundant).

The *backtracking* line search method only requires that

$$\overline{J^{(T)}}\left(K^{(i)} - a^{(i)}(d\overline{J^{(T)}}/dK)^{(i)}\right) \leq \overline{J^{(T)}}(K^{(i)}) \tag{4.14}$$

is satisfied on each iteration. It's implementation is straightforward:

Algorithm 5: Backtracking

1. Set $j = 0$. Choose a value $a^{(j=0)}$ and $c \in [0, 1]$.
2. Integrate $\dot{\mathbf{u}} - \mathbf{h}(\mathbf{u}, K^{(i)} - a^{(j)}(d\overline{J^{(T)}}/dK)^{(i)}) = 0$. If $\overline{J^{(T)}}(K^{(i)} - a^{(j)}(d\overline{J^{(T)}}/dK)^{(i)}) \leq \overline{J^{(T)}}(K^{(i)})$ is satisfied, break and use the step size $a^{(i)} = a^{(j)}$. If the condition is not satisfied, set $a^{(j+1)} = ca^{(j)}$.
3. Set $j = j + 1$ and return to Step 2.

The time-averaged quantity $\overline{J^{(T)}}$ needs to be ergodic (i.e. independent of the initial condition), as explained in Section (2.2). It must also be noted that the nonlinear objective function $J(\mathbf{u}, K)$, and therefore $\overline{J^{(T)}}(\mathbf{u}, K)$, are generally non-convex with respect to the elements of K [93]. Therefore in general, convergence to the global minimum is not guaranteed.

Günther et al. [23] also proposed an optimisation algorithm for chaotic systems which uses shadowing. They used a *single-step one-shot* approach which replaces the nonlinear constraint of the minimisation statement with the solution to the well-conditioned nonlinear shadowing (2.2) problem. They applied the algorithm to a tracking control problem of an advection-dominated flow in one dimension. The sensitivities were used to update a single control parameter governing the boundary condition using a quasi-Newton approach (BFGS). In this chapter, however, all elements of the feedback matrix; 255^2 in total, are computed. The current approach also bypasses the need to store and solve the large Schur complement system (2.7), which makes the approach of Günther et al. [23] too difficult to apply to large systems. Algorithm (4) only relies on preconditioned matrix-vector products to solve (4.13), making the control Algorithm (3) more suitable for application to large chaotic systems.

4.4 Control of the Kuramoto Sivashinsky Equation

In this section, the two control approaches outlined in Section (4.3) are applied to the standard Kuramoto Sivashinsky equation (KSE), repeated below for convenience,

$$\begin{aligned}
&\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^4 u}{\partial x^4} \quad x \in [0, L] \\
&u(0,t) = u(L,t) = 0 \\
&\left.\frac{\partial u}{\partial x}\right|_{x=0} = \left.\frac{\partial u}{\partial x}\right|_{x=L} = 0
\end{aligned} \tag{4.15}$$

The Dirichlet and Neumann boundary conditions ensure ergodicity of the system, and for all simulations, $L = 128$. The spatial derivatives used the same finite difference discretisation shown in Appendix (A) and used previously in Section (3.5.2). Two values of node spacing $\delta x = {}^{L}/_{(N+1)}$ were considered, 1 and 0.5. The variable step Runge–Kutta method (ode45 in MATLAB) was used again for the integration in time. The initial condition at $t = 0$ was obtained from a precursor integration in $-1000 \leq t \leq 0$; this ensured that the trajectory had reached the chaotic attractor at $t = 0$.

Figure (4.1) shows the typical streaky behaviour exhibited by the KSE in the $x - t$ plane. It is clear that there is a characteristic average streak spacing, l_{str}, with wavenumber $k = {}^{2\pi}/_{l_{str}}$. As will be shown later, l_{str} plays an important role in the analysis of control kernels. If the boundary conditions are periodic, the spectral energy peaks at k. This characteristic value is close to the wavenumber that maximises the eigenvalue of the linearised KSE around the rest state. For periodic boundary conditions, this corresponds to a streak spacing that can be found analytically as $l_{str}^{per} = 2\pi\sqrt{2}$ [64], which is equal to $l_{str}^{per} = 8.9$. As will be seen later for the system (4.15), $l_{str} \approx 8.5$, therefore the choice of the boundary conditions does not affect much the streak spacing.

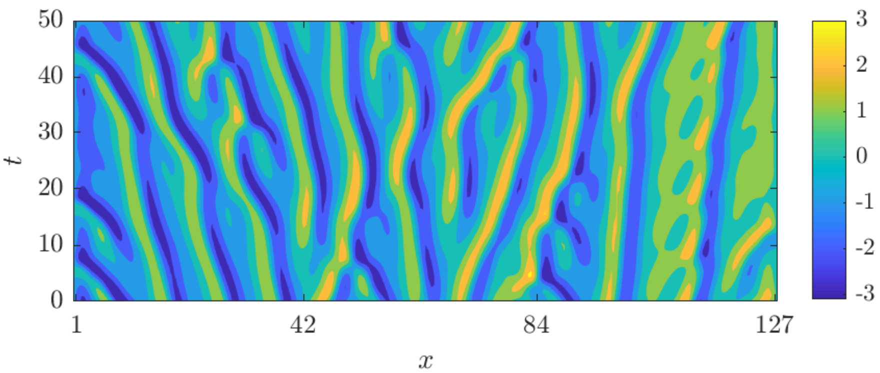
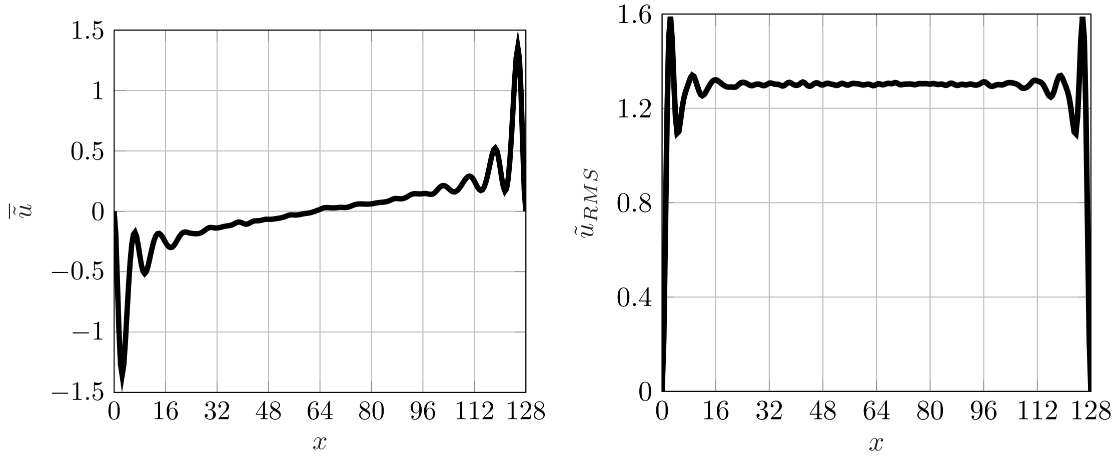


Figure 4.1: Contour plot of a typical solution $u(x, t)$ of (4.15).

The time-average and root mean square (rms) of u as a function of x, $\overline{\tilde{u}}$ and $\tilde{u}_{RMS}$

respectively, are shown in Figure (4.2). The overbar $\bar{\cdot}$ denotes time-averaging, while $\tilde{\cdot}$ represents averaging over multiple initial conditions $\mathbf{u}_0$. The Dirichlet boundary conditions result in sharp gradients at both ends of the domain, especially for $\tilde{u}_{RMS}$. In the middle of the domain, the variation of both the mean and rms is smooth.



(a) Time average of $u(x,t)$

(b) Temporal RMS of $u(x,t)$

Figure 4.2: Time average and RMS of $u(x,t)$. $\bar{\tilde{u}}$ and $\tilde{u}_{RMS}$ were obtained for trajectories with length $T = 2000$ and averaged over 150 random initial conditions in $[0,1]$.

The discretised KSE equation is written as

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}) - BK\mathbf{u} \tag{4.16}$$

where $\mathbf{f}(\mathbf{u})$ is the nonlinear vector arising from the finite difference discretisation of the right-hand-side of (4.15) (as shown in Appendix A), $\mathbf{u}(t) = \begin{bmatrix} u_1 & u_2 & \dots & u_N \end{bmatrix}^T$ and K is the feedback matrix. For shadowing control, the optimal values of K that minimise

$$\overline{J^{(\infty)}} = \lim_{T\to\infty} \frac{1}{TL} \int_0^T \left(\mathbf{u}^T \delta x \mathbf{u} + \frac{\alpha}{2} \left(K\mathbf{u}\right)^T \delta x \left(K\mathbf{u}\right) \right) dt \tag{4.17}$$

subject to the nonlinear constraint (4.16), are sought. The objective and constraint have the same form as (4.10). The first term on the right-hand-side of (4.17) represents the space-time average kinetic energy of the system, while the second term represents the cost of the control effort, which is regulated by the parameter α. It is assumed that there are N equally spaced actuators (equivalent to setting $B = I$ in equation 4.16). The aim is to find the optimum values of the elements of K (which is size $N \times N$) that drive $\mathbf{u}(t)$ to zero, i.e. that bring the system to rest.

In the following section, this matrix K is compared with the one obtained using LQR. For the latter, the linear version of the KSE (4.15), i.e.

$$\frac{\partial u}{\partial t} = -\frac{\partial^2 u}{\partial x^2} - \frac{\partial^4 u}{\partial x^4} \quad x \in [0, L] \tag{4.18}$$

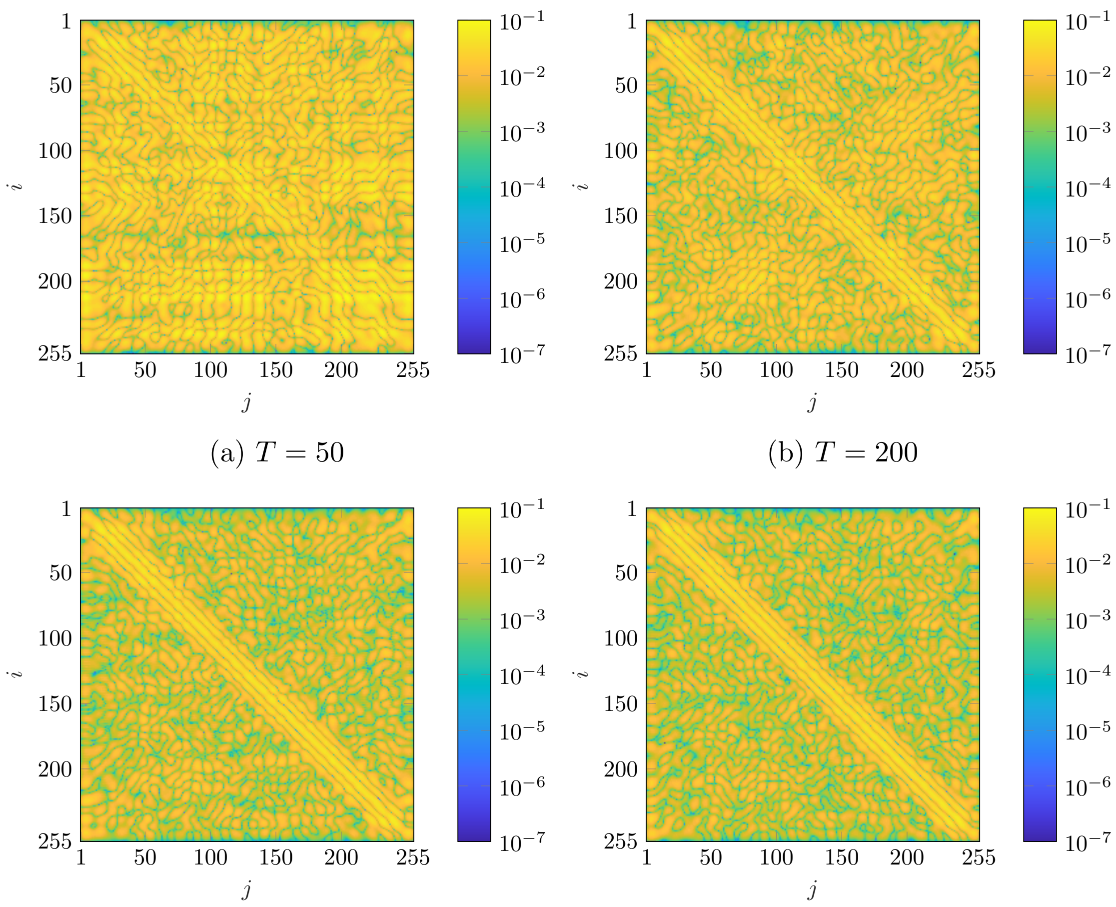
with the same boundary conditions, is discretised to obtain the linearised matrix A_l, which is required to compute the LQR feedback matrix (4.9). Both matrices are then applied to the full nonlinear discretised KSE (4.16).

4.4.1 Comparison of the PMSS and LQR control kernels

Since the input matrix B has been set to $B = I$, the actuation $\mathbf{s}(t) = -K\mathbf{u}(t)$ can be written explicitly as $s_i(t) = -\sum_{j=1}^{N} K_{i,j} u_j(t)$, where the index i corresponds to the control location, $x_c = i \times \delta x$. Written in this form, the physical meaning of $K_{i,j}$ becomes clear; it represents the weight of the $j-th$ velocity to the actuation at the $i-th$ point. Summing all j contributions results in $s_i(t)$. For more general cases, the input matrix B provides an appropriate spatial weighting, and the control signal is given by $B\mathbf{s}$, see equation (4.1). The total derivative of $\overline{J^{(T)}}$ with respect to element $K_{i,j}$ is found using (2.34). For the objective function (4.17) considered here, (2.34) translates into

$$\frac{d\overline{J^{(T)}}}{dK_{i,j}} = -\left(\sum_{p=1}^{P} \int_{t_{p-1}}^{t_p} \big(u_j(t) \hat{w}_i(t) \big) \, dt \right) + \frac{\alpha \delta x}{TL} \int_0^T \left(u_j(t) \sum_{n=1}^{N} u_n(t) K_{i,n} \right) dt \tag{4.19}$$

where $\hat{w}_i(t)$ are the adjoint variables obtained by integrating (2.32). Figure (4.3) shows the absolute values of the elements of the sensitivity matrix $\left| d\overline{J^{(T)}} / dK_{i,j} \right|$ in log-scale, for different trajectory lengths T. For $T = 50$ (Panel a), the matrix does not seem to have a clear structure. For $T = 200$ (Panel b), the matrix is starting to acquire a diagonally dominant structure. However, a significant number of elements in the top right and bottom left quadrants are of the same order of magnitude $O(10^{-1})$ as the central diagonals (shown in yellow). The time horizons $T = 50$ and $T = 200$ are clearly not long enough for sensitivity convergence. However, by increasing the trajectory length to $T = 500$ (Panel c) and $T = 800$ (Panel d), the sensitivities are starting to converge and become independent of T. It is now clear that the matrix $\left|d\overline{J^{(T)}}/dK\right|$ has a diagonally dominant structure. Note that convergence is first attained around the main diagonal, and slowly propagates



(a) $T = 50$ (b) $T = 200$

(c) $T = 500$ (d) $T = 800$

Figure 4.3: Colour maps of the absolute values of sensitivities $\left|d\overline{J^{(T)}}/dK\right|$ in log-scale for different time-averaging lengths T. They were obtained from the attractor of the uncontrolled system (i.e. the first iteration of Algorithm (3) with $K^{(0)} = 0$).

further away.

Using the computed sensitivities for $T = 800$ and a step size $a^{(0)} = 10$, the matrix $K^{(1)}_{i,j}$ computed from (4.12) was plotted in a colour map in Figure (4.4). Large positive and negative values are found around the main diagonal, while further away, i.e. when $j \gg i$, $K_{i,j}$ decays to 0. This has a clear physical meaning; the value of actuation at point i is determined mainly by nearby neighbours, while the contribution of points located further away becomes progressively smaller and smaller. The control kernel therefore has compact support. This has been demonstrated for the kernel obtained when LQR is applied to the linearised Navier-Stokes equations in a channel flow (see Figure (6) of [94]). The present analysis shows that the same property holds if K is computed directly from the nonlinear attractor, at least for the case examined. More research is needed however, to determine if controllers with compact support can be computed for general nonlinear systems using the current algorithm.

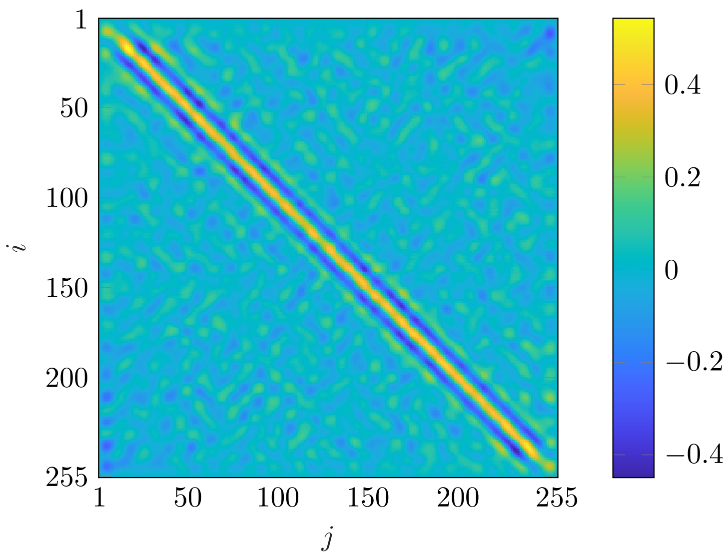


Figure 4.4: Colour map of matrix K obtained using PMSS control with $T = 800$.

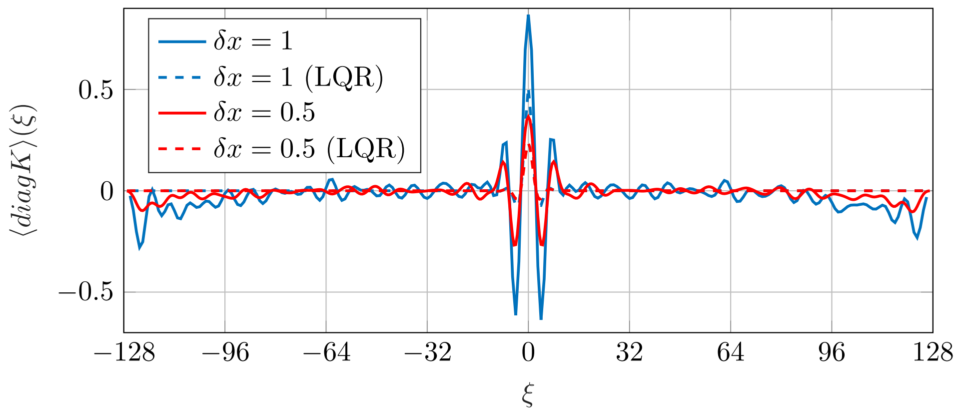
Averaging along the m-th diagonal of K (the main diagonal corresponds to $m = 0$), $\langle diagK\rangle(\xi)$ is obtained, which depends only on $\xi = m\delta x = x - x_c$. It is clear that $\langle diagK\rangle(\xi)$ represents the average value of weights against the distance ξ from the actuation point. The distribution of $\langle diagK \rangle(\xi)$ is plotted for two grid spacings δx in Figure (4.5a) with solid lines. The weights obtained from the LQR matrix with $c = 1$ (refer to equation 4.9) are superimposed with dashed lines. Notice that the weights obtained from LQR and PMSS have very similar distributions. As expected, they are both localised around $\xi = 0$ and decay to 0 further away. They also both depend on the discretisation.

In order to further analyse the results, the control kernel is computed, which is independent of δx. To this end, the actuation at location x_c is written as the convolution integral

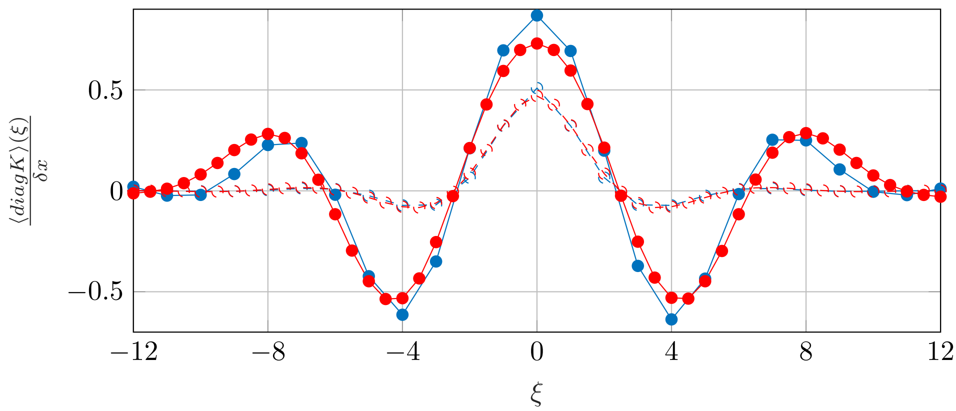
$$s(x_c) = -\int_0^L \mathscr{K}(x - x_c; x_c)\, u(x)\, dx \tag{4.20}$$

where $\mathscr{K}(x - x_c; x_c)$ is the convolution (or control) kernel at x_c. This can be computed from the matrix elements $\mathscr{K}(x - x_c; x_c) = \mathscr{K}(\xi; x_c) = {}^{K_{i,j}}/_{\delta x}$, where $x_c = i \times \delta x$ and $x = j \times \delta x$. In Figure (4.5b), both diagonally-averaged kernels are plotted, and in order to facilitate the comparison, the region $\xi \in [-12, 12]$ is zoomed into. The LQR kernels (dashed lines) collapse perfectly for the two discretisations, as they should. The PMSS kernels however (solid lines), although close, do not collapse to a single curve. Perhaps they should not be expected to collapse, as they are found directly from the nonlinear attractor under the assumption of the feedback control law (4.2). The shapes produced by the two control methods are similar, but the LQR kernel decays to 0 faster than the PMSS kernel. The latter has more pronounced peaks and oscillates around 0.

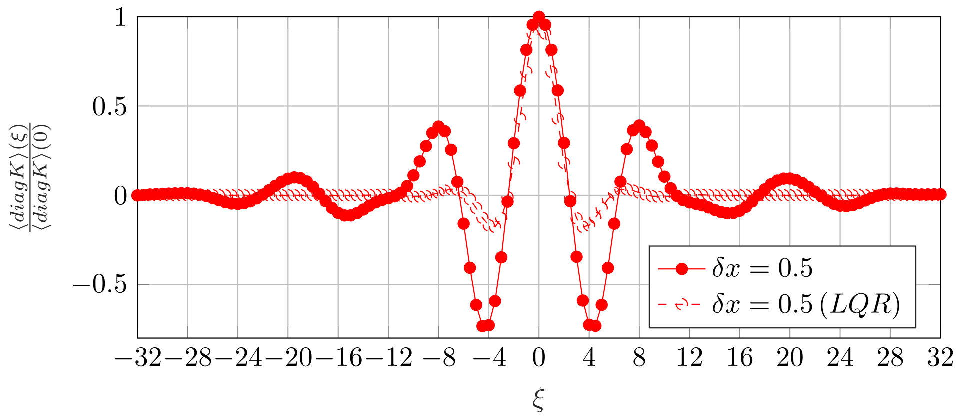
To eliminate the effect of the step size $a^{(0)}$ and the control cost parameter c that affect the absolute values of $\overline{\mathscr{K}}(\xi)$, the kernel distribution normalised with the value at $\xi = 0$ is plotted in Figure (4.5c). This normalisation reveals a very interesting feature; the two kernels are almost identical in the region $-2.5 \leq \xi \leq 2.5$, but deviate elsewhere.



(a) Full domain



(b) Mean convolution kernel in the region $-12 \leq \xi \leq 12$



(c) Mean convolution kernel normalised by ${}^{\langle diagK \rangle(0)}/_{\delta x}$

Figure 4.5: Distribution of the feedback matrix weights (Panel a) and kernels (Panels b,c), obtained by averaging along the diagonals of K and plotting against $\xi = x - x_c$.

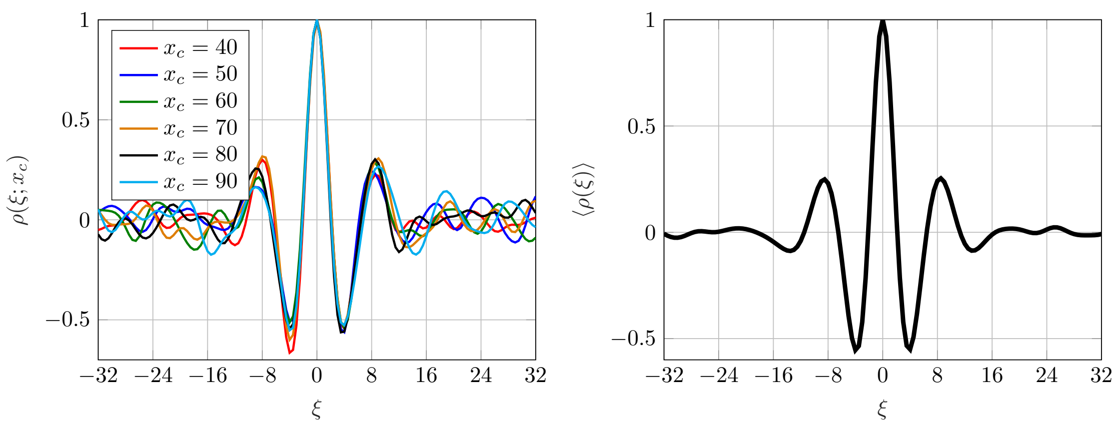
What determines the shape of the kernels and why do they have this distribution?

In order to gain more insight, the two-point spatial correlation function at a given location x_c is considered:

$$\rho(\xi; x_c) = \frac{\overline{u'(x_c, t)u'(x_c + \xi, t)}}{\overline{u'(x_c, t)^2}} \tag{4.21}$$

where $u' = u - \overline{u}$ is the fluctuation about the time-average $\overline{u}$. A small correlation $\rho(\xi; x_c) \approx 0$ indicates that a perturbation at $x_c + \xi$ (for example due to actuation), would not be 'seen' at x_c. It is therefore expected that this function will be related to the control kernel.

The correlation $\rho(\xi; x_c)$ is plotted against ξ at different locations x_c along the domain in Figure (4.6a). The correlations collapse very well for $\xi \in [-4, 4]$, but start to deviate as ξ becomes larger. The fluctuations of $\rho(\xi; x_c)$ around zero that occur for large $|\xi|$ are due to finite time-averaging, and decay very slowly to zero at $T \to \infty$. Note also the slight loss of symmetry around $\xi = 0$ for points x_c close to the boundaries of the domain. The correlation $\rho(\xi; x_c)$ is then averaged over x_c in the region $40 < x_c < 90$, and the distribution of the spatially averaged correlation $\langle \rho(\xi) \rangle$ is then plotted in Figure (4.6b). Symmetry around $\xi = 0$ has now been restored. Distinct positive and negative peaks can be identified at $\xi \approx \pm 4, \pm 8.5, \pm 13.5$ etc. Moreover, $\langle \rho(\xi) \rangle$ decays to zero for $\xi < -20$ and $\xi > 20$.



(a) Correlations $\rho(\xi; x_c)$ against ξ for different x_c

(b) Averaged correlation in the region $x_c \in [40, 90]$

Figure 4.6: Two-point spatial correlations.

These results can be explained by reference to Figure (4.1). The peaks at $\xi = \pm 4$ indicate the average distance between positive and negative streaks that are located next to each other. The fluctuations around the average $\overline{u}$ have opposite signs and therefore $\langle \rho(\pm 4) \rangle < 0$. The lower peaks at $\xi = \pm 8.5$ indicate a weaker correlation between two positive or two negative streaks. The even lower peaks at $\xi = \pm 13.5$ can be explained similarly. The results indicate that the correlation is strong over approximately $\xi = 8.5$, i.e. over the average distance between streaks of the same sign.

In Figure (4.7), the spatially averaged correlation $\langle \rho(\xi) \rangle$ is plotted together with the normalised LQR and PMSS control kernels. Plotted in this way, all three distributions have remarkable similarities, but also some differences. The two control kernels and the correlation overlap in the region $-2.5 \leq \xi \leq 2.5$. At $\xi = \pm 4$, the troughs are more clearly pronounced for the PMSS kernel and are closer to $\langle \rho(\pm 4) \rangle$. On the other hand, the LQR kernel decays very quickly outside the ξ region $[-4, 4]$. The narrow support of the LQR kernel indicates that the actuation acts to annihilate the positive/negative streaky combinations. On the other hand, the PMSS kernel has wider support and opposes larger positive/negative/positive streaky combinations.

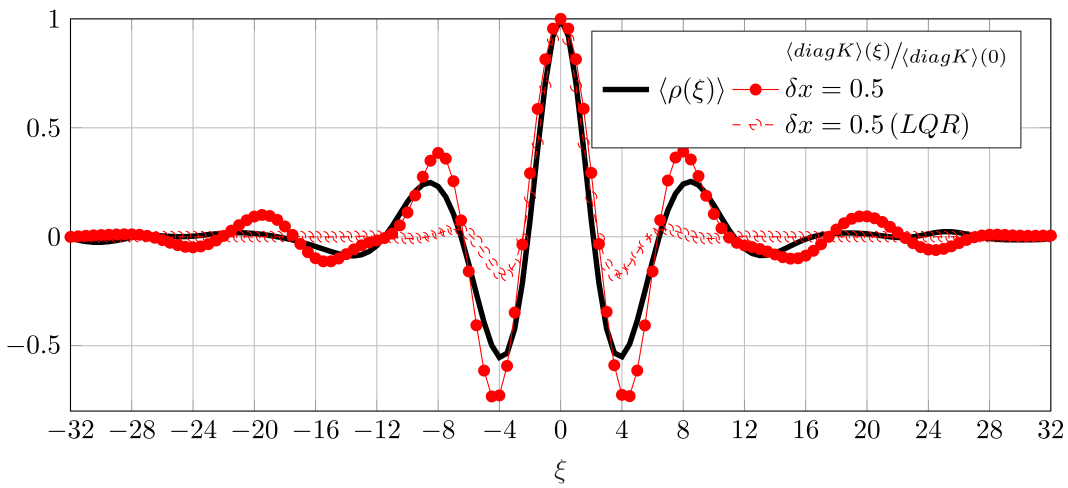


Figure 4.7: The spatially averaged two-point correlation $\langle \rho(\xi) \rangle$ superimposed on ${}^{\langle diagK \rangle(\xi)}/_{\langle diagK \rangle(0)}$.

4.4.2 Response of the controlled system

The response of the system (4.16) to PMSS and LQR actuation is now discussed. The instantaneous, spatially-averaged kinetic energy,

$$J(t) = \frac{1}{L}\left(\mathbf{u}(K,t)^T \delta x \mathbf{u}(K,t)\right) \tag{4.22}$$

is used to determine which controller stabilises the system fastest. For PMSS, control with different kernel sizes are used, i.e. a value ξ_{max} is chosen, and the actuation is computed using only the elements of K in the region $-\xi_{max} \leq \xi \leq \xi_{max}$ around each control point x_c. The step size $a^{(0)} = 6$ is set on Step 5 of Algorithm (3). To ensure a fair comparison, a value $c = 1.1$ in equation (4.9) was found (by trial and error) so that the average of the main diagonal, $\langle diag(K)\rangle(0)$, is the same for the PMSS and LQR controllers.

The energy $J(t)$ is plotted against time in Figure (4.8). For $a^{(0)} = 6$, all controllers bring the system to rest and so no additional iterations are required. It is remarkable that only a single PMSS control iteration, which relies only on information of the uncontrolled system, is so effective. When $\xi_{max} = 2$, the normalised LQR and PMSS matrix kernels are very similar (as shown in Figure 4.7) and hence $J(t)$ drops at the same rate for both controllers. A contour plot of the controlled solution is shown in Figure (4.9), which shows how effective the actuation is. It can be clearly seen that the streaks found in the uncontrolled case (Figure 4.1) are rapidly annihilated upon application of the control, and $|u(x,t)|$ is brought to 0 within $5-10$ time units.

Increasing ξ_{max} to 20 reduces the rate at which the system is brought to rest. Using the full matrix, i.e. setting $\xi_{max} = 127$, still reduces the instantaneous energy by 4 orders of magnitude (from $J(0) = 1.68$ to $J(T) = 4.8\times10^{-4}$). However, the response of the system is slow when $J(t)$ falls below 10^{-2}, i.e. this is a long-term effect.

Indeed, as can be seen from Figure (4.8), and especially from the inset that zooms-

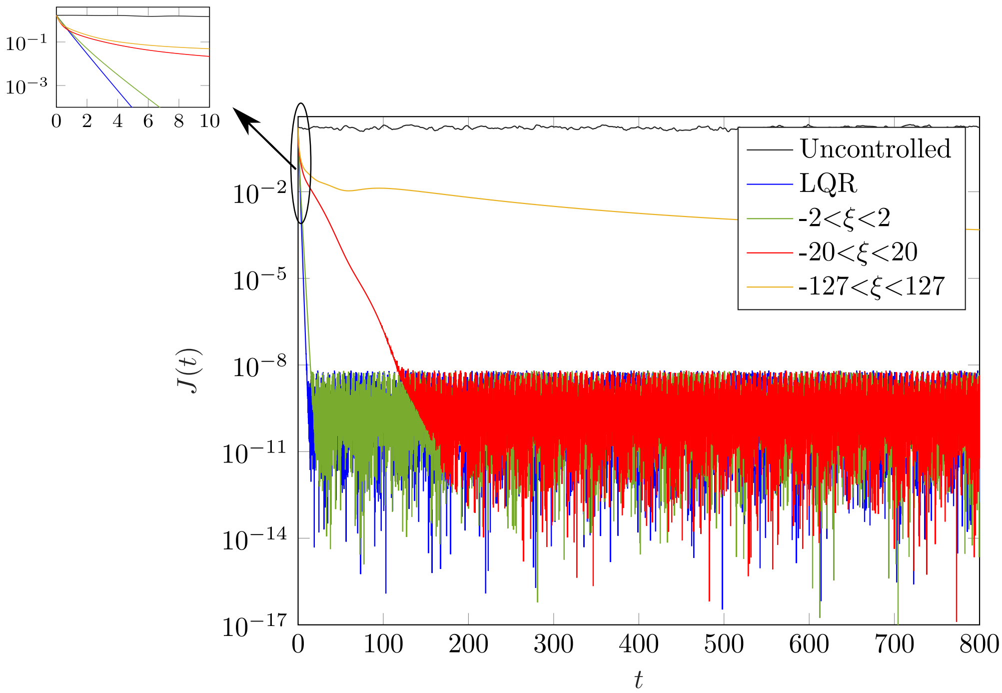


Figure 4.8: Instantaneous kinetic energy $J(t)$ of the actuated system using PMSS and LQR. The uncontrolled case is shown in black colour and the LQR in blue. For the PMSS control matrix K, only the elements that fall inside the indicated range of ξ are used. Decreasing ξ_{max} led to faster stabilisation for the values considered.

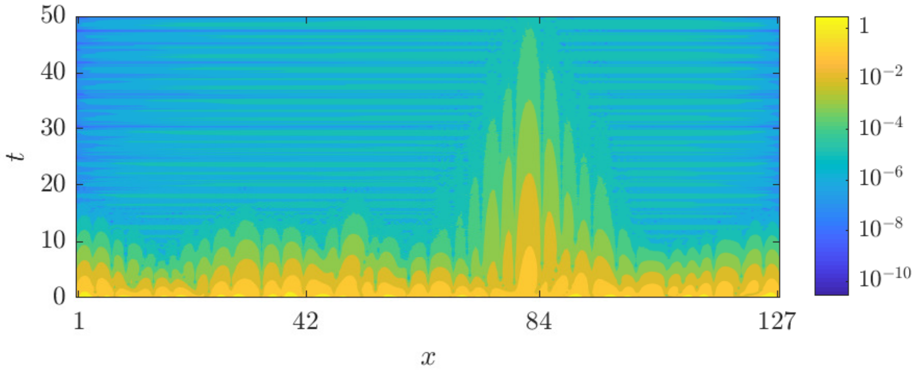
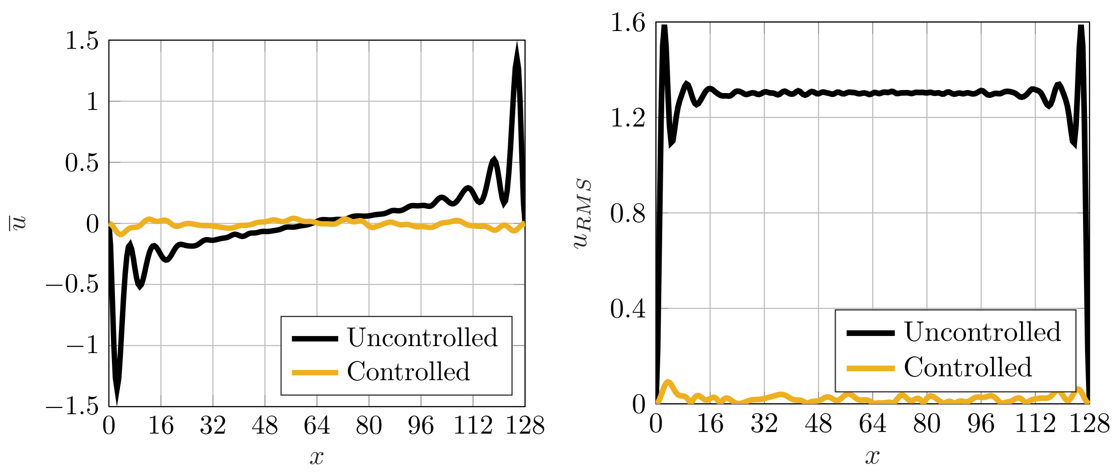


Figure 4.9: Absolute values of the controlled solution $|u(x,t)|$ in log-scale (with $\xi_{max} = 2$).

in to small values of t, the rate of descent is initially the same for PMSS and LQR. Most importantly, for the PMSS controller, this holds for all values of ξ_{max}, even the largest. The curves start to deviate for approximately $t > 1$. In Figure (4.10), the spatial distribution of the time-average and rms of $u(x,t)$ are plotted with and without control (the results are with $\xi_{max} = 127$). Note the effectiveness in suppressing both variables and bringing them close to 0 across the whole domain.



(a) Time average of $u(x, t)$

(b) RMS of $u(x, t)$

Figure 4.10: Time average and RMS of $u(x, t)$. The controller $K^{(1)}$ used the full matrix, i.e. with $-127 \leq \xi \leq 127$. The statistics were computed by time-averaging between $t = 500$ and $t = 800(= T)$.

In order to shed more light onto the behaviour shown in Figure (4.8), the matrix $A_l - K^{(1)}$ is considered, where A_l is the discrete form of the right-hand-side of the linearised equation (4.18). It is expected that for the controlled systems, the eigenvalue of $A_l - K^{(1)}$ with the largest real part will determine how fast the system is stabilised. The nonlinear term $u{\partial u}/{\partial x}$ is suppressed with actuation, as shown in Figure (4.11), therefore the eigenvalues $\mu\left(A_l - K^{(1)}\right)$ are sufficient to determine the stability of the controlled systems.

Figure (4.12) shows the eigenvalues, μ, for two values of $\xi_{max} = 2, 127$. Note that for the uncontrolled case, all $\mu(A_l)$ are real because the matrix A_l is symmetric (contains only second and fourth-order derivatives that are discretised with central differences). However, some $\mu(A_l - K^{(1)})$ have a small imaginary part due to the lack of strict symmetry of $K^{(1)}$. This is likely due to the finite length of the trajectory T and the slow convergence of the off-diagonal elements. Some eigenvalues of A_l have positive real parts, indicating linear instability. When the controller is introduced, all eigenvalues move to the left-hand plane, i.e. the system becomes stable. However,

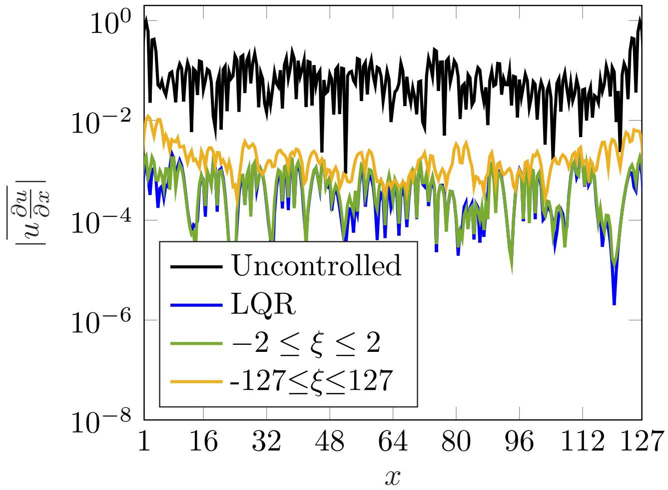
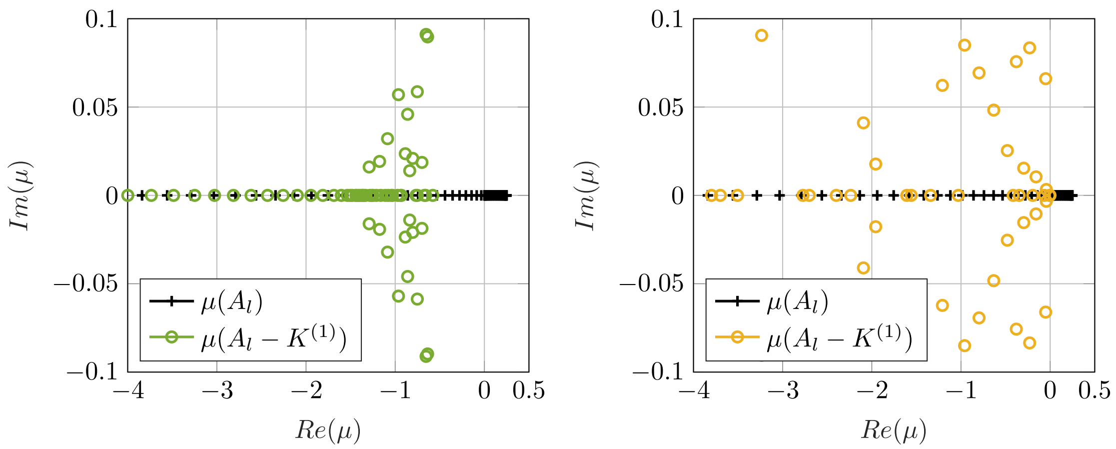


Figure 4.11: Comparison of the time-averaged absolute values of the nonlinear term $|\overline{u\,^{\partial u}/_{\partial x}}|$ using different actuations.



(a) $K^{(1)}$: $-2 \leq \xi \leq 2$

(b) $K^{(1)}$: $-127 \leq \xi \leq 127$

Figure 4.12: Eigenvalues of the controlled and uncontrolled matrices, $A_l - K^{(1)}$ and A_l, respectively, plotted in the complex-plane ($K^{(1)} = -a^{(0)} \, ^{d\overline{J(T)}}/_{dK^{(0)}}$ for $T = 800$ and $a^{(0)} = 6$). Only values with $Re(\mu) > -4$ are shown.

for $\xi_{max} = 127$, some eigenvalues are close to the imaginary axis, explaining the slower rate of descent of $J(t)$.

4.5 Algorithm Performance

The performance of Algorithm (3) is now discussed. The inputs $T = 50$, $K^{(0)} = 0$, and $\epsilon = 1 \times 10^{-2}$ (stopping criteria) were set, while restricting K to the diagonals in the region $-2 \leq \xi \leq 2$. A number of performance measures are given in Table (4.1). It is clear that a single iteration is sufficient to drive the kinetic energy evaluated at $t = T$, $J(T)$, to zero. Since one iteration renders the controlled system linearly stable, i.e. $Re\big(\mu_{max}\big(A_l - K^{(1)}\big)\big) < 0$, a second iteration is unnecessary.

Iteration #	$\overline{J^{(T)}}$	$J(T)$	$\Vert d\overline{J^{(T)}}/dK^{(i)}\Vert_2$	$a^{(i)}$
$i = 0$	1.62	1.37	3.7×10^{-1}	10
$i = 1$	1.4×10^{-2}	$\approx 1 \times 10^{-10}$	1×10^{-2}	10
$i = 2$	1.4×10^{-2}	$\approx 1 \times 10^{-10}$	-	-

Table 4.1: Some key PMSS control (Algorithm 3) performance measures. The algorithm inputs used are $T = 50$, $K^{(0)} = 0$ and $\epsilon = 1 \times 10^{-2}$. K is restricted to $-2 \leq \xi \leq 2$.

The time-average over $[0, T]$, $\overline{J^{(T)}}$, is reduced by 2 orders of magnitude between the $0-th$ iteration (uncontrolled flow) and the controlled flow after one iteration. The actual value of $\overline{J^{(T)}}$ depends on T and accounts for the transient shown in Figure (4.8). The variation of $\overline{J^{(T)}}$ against T (when the algorithm is run separately for each T) is shown in Figure (4.13). $\overline{J^{(T)}}$ drops with rate $\sim T^{-1}$, because the transient period occupies a smaller and smaller fraction of T as $T \to \infty$. The 2-norm of the sensitivity matrix $\Vert d\overline{J^{(T)}}/dK^{(i)}\Vert_2 = \sigma_{max}\big(d\overline{J^{(T)}}/dK^{(i)}\big)$, where σ_{max} denotes the maximum singular value, drops by a factor of three from the $0-th$ to the 1^{st} iteration. Again, this depends on T.

Table (4.2) shows the Adjoint PMSS solver (Algorithm 4) parameters and performance (Algorithm 4 is called on Step 3 of Algorithm 3). Initially, the uncontrolled trajectory ($K^{(0)} = 0$) has 15 positive Lyapunov exponents ($N_{+LE} = 15$). The preconditioner therefore uses $l = 15$ on the first iteration (to annihilate these fastest growing modes). The condition number κ is reduced by four orders of magnitude, and the convergence of GMRES is achieved in 34 iterations. On the second iteration, all unstable modes have been annihilated ($N_{+LE} = 0$), and therefore no preconditioning is necessary (the condition number $\kappa(S) = 5$ is already very small).

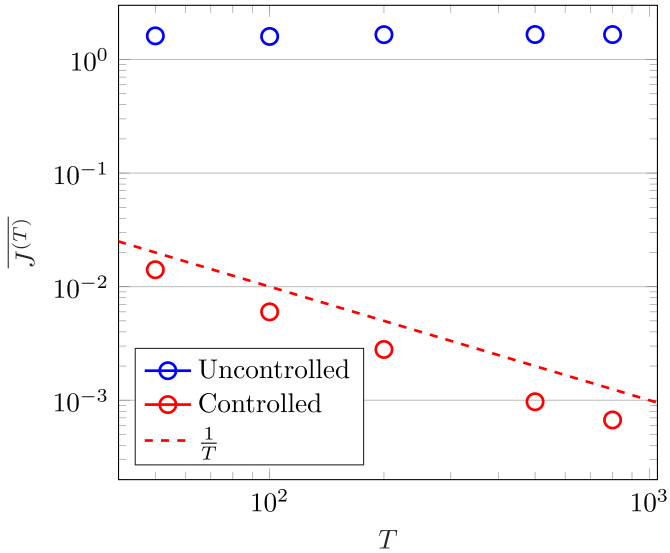


Figure 4.13: Space-time averaged objective $\overline{J^{(T)}}$ obtained by running Algorithm (3) for different T; it drops with rate $\sim T^{-1}$.

Iteration #	N_{+LE}	ΔT	l	q	γ	$\kappa(S)$	$\kappa(H)$	# GMRES Iterations
$i = 0$	15	5	15	1	0.1	1.3×10^5	24	34
$i = 1$	0	5	-	-	0.1	5	-	2

Table 4.2: Adjoint PMSS solver (Algorithm 4) parameters. N_{+LE} refers to the number of positive exponents of the trajectory on Step 3 of Algorithm (3). $\kappa(S)$ and $\kappa(H) = \kappa\left(\gamma I + \mathrm{M_{BD}}{}^{(q)}_{(l)} S\right)$ are the condition numbers of the unconditioned and preconditioned MSS matrices, respectively.

For Algorithm (3) to be applicable to large systems, it must be able to compute accurate sensitivities as efficiently as possible. The solution to the linear system (4.13) dominates the computational cost of the algorithm, making preconditioning necessary for scalability. In Figure (4.14), the convergence rate, quantified in terms of the GMRES residuals ($\|r_{(m)}\|_2 = \|\underline{q} - H\underline{\hat{\mathbf{w}}}_{(m)}\|_2$), is plotted against the GMRES iteration number m, where $\underline{q}$ and H are, respectively, the right-hand-side vector and matrix of (4.13). It is clear that the dependence of the convergence rate on T is very weak (as previously shown for tangent MSS in Figure (3.17a) for the KSE). The residuals drop by 5 orders of magnitude in between 25-37 iterations, even though T varies by more than an order of magnitude, from $T = 50$ to $T = 800$, increasing the number of unknowns $\underline{\hat{\mathbf{w}}}$ by a factor of 16.

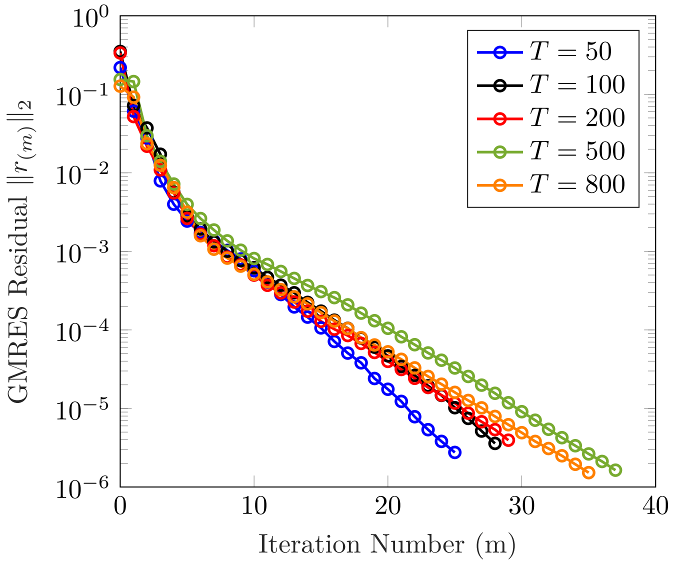


Figure 4.14: GMRES Residuals $\|r_{(m)}\|_2$ for Step 3 of the Adjoint PMSS solver (Algorithm 4). The preconditioner parameters used are $\Delta T = 10$, $l = 15$, $q = 1$ and $\gamma = 0.1$. The residual $\|r_{(m)}\|_2$ drops by approximately five orders of magnitude by the final iteration for all T values.

4.6 Summary

An algorithm that couples shadowing adjoint sensitivity analysis with gradient descent was proposed to simultaneously compute all the elements of the feedback control matrix K for chaotic systems. The sensitivities were used as search directions to find the matrix elements that minimise an objective function, subject to the full nonlinear system constraints. Most importantly, due to the adopted adjoint formulation, the computational cost is independent of the number of the matrix elements. Algorithm (3) was applied to control the KSE, and the actuated system was successfully stabilised around the rest position, $\mathbf{u}_{targ} = 0$. It was shown that for suitably chosen parameters, a single iteration of the algorithm was sufficient to compute a stabilising feedback matrix K.

The control kernels obtained using Algorithm (3) were compared with the standard LQR kernels. Similarities and differences were noted. Both kernels had compact support and similar shape, which was related to the streaky structure of the solution of the uncontrolled KSE. A very similar control kernel shape can be expected for longer domain lengths L, since the streak spacing is independent of L. They were almost identical for short separations from the actuation point, but the LQR kernel decayed faster to 0. This difference is most likely due to the nonlinear terms that are ignored in LQR. All kernels computed with Algorithm (3) were stabilising.

From a computational point of view, the cost of LQR scales with $O(N^3)$, which poses severe restrictions for applications to large systems. On the other hand, the PMSS algorithm uses only time steppers and a preconditioner to make the linear solver convergence rate almost independent of N and T. However, as discussed in Section (3.8), the cost of preconditioning increases with N_{+LE}, so for the case of the KSE, the preconditioning cost would likely increase with L. While the LQR showed faster stabilisation of the instantaneous energy for the case examined (see Figure 4.8), the performance of the PMSS controller was almost identical for the restricted kernel.

Chapter 5

State Reconstruction of Chaotic Systems from Limited Measurements

5.1 Introduction

The acceleration approach proposed in Chapter (3) is applicable whenever a minimisation problem requiring the application of the multiple shooting method needs to be solved. In the previous chapter, it was applied to compute the elements of a feedback matrix. In this chapter, the state reconstruction of chaotic systems from limited measurements is considered. As will be in the next section, this can be formulated as an optimisation problem for which the acceleration approach of Chapter (3) can be applied.

Measurements of real physical systems are usually taken at limited points in space and time, which are usually corrupted by noise. Additionally, the mathematical description of the physical system may be incomplete and subject to unknown dynamics and uncertainties, such as unknown initial conditions and parameter values. It is therefore important to fuse the experimental data with uncertain dynamical models in order to estimate the true system behaviour. This process is called *data assimilation*. The aim is to reconstruct all the system states from limited measurement data while minimising discrepancies between the measured data and the predictions of the uncertain model. Data assimilation techniques can be broadly categorised into *variational* and *sequential* methods. Sequential methods such as Kalman filtering produce an improved estimate every time a new measurement is available. On the other hand, variational methods operate over a broader time window, i.e. consider a larger number of measurements in time. For variational (VAR) methods, an objective function quantifying the mismatch between the available measurements and their estimation from the model over a given time window, is minimised.

The common VAR approach [12, 13, 14, 15] requires integration of the non-linear equations (1.1) and backward-in-time integration of the adjoint equations, to compute the derivatives of the objective w.r.t the control parameters (usually uncertain system parameters and/or the initial conditions). Using a descent algorithm (such as steepest descent or conjugate gradient), the control values that minimise the objective are found iteratively. The associated storage costs are usually low, and parallel integration of the non-linear and adjoint equations can be used to reduce the computing time [95].

This approach has a few drawbacks, however. Slow convergence can be expected if real measurements are used [96]. Since each iteration requires an integration of the non-linear and adjoint equations, it is crucial to minimise the iteration count. Preconditioned Quasi-Newton updates are often used to accelerate the convergence [96], but these require the computation and storage of an approximate Hessian matrix. Finding the control values that minimise the objective is straightforward if the objective is convex w.r.t the control values. Systems undergoing transition may have two minima, and convergence to a local minimum may occur, depending on the initial control values [12].

For chaotic systems, such an approach may only be useful for short time assimilation windows T [97, 13], due to the exponential growth of initial value errors. Hence for chaotic systems, it may be better to compute incremental updates for segmented time windows [96, 98]. The idea is to use linear operators to compute updates of the estimation (and controls) at discrete points in time. In [98], the discrete-time updates were obtained by solving a preconditioned saddle point system.

There is much scope for improvement with regards to segmented VAR data assimilation for chaotic systems. In order to accelerate the convergence, the preconditioning of the saddle point system arising from the solution to the corresponding optimisation problem is one area which requires particular attention. In this chapter, an iterative VAR algorithm is derived, which uses the Newton-Raphson method to solve an optimisation problem. At each outer iteration, a saddle point system similar to that derived in [98] must be solved iteratively (inner iterations). The preconditioner introduced in Chapter (3) is also used here (with slight modifications, as will be shown in Section 5.3.1), to accelerate the convergence rate of the Schur complement system solution. Results are presented for the Lorenz system to demonstrate the efficacy and accuracy of the proposed method.

5.2 Problem Formulation

In practical applications, only some system states (or functions thereof) can be measured. In this section, a method to reconstruct all the states of a chaotic system from such limited measurements is presented. A similar KKT system to that derived for MSS (equation 2.23) is also derived here, and it is shown that the preconditioner proposed in Section (3.4) can be used with minor modifications. As will be shown later, the preconditioner results in a significant acceleration of the solution of the resulting linear system.

Suppose that a set of states $\mathbf{u}_m(t)$ (assumed continuous and possibly noisy) have been obtained experimentally. The vector $\mathbf{u}_m(t)$ has length N, and unmeasured states are replaced by zeros (i.e. $\mathbf{u}_{m,i} = 0$ if the i^{th} state is not measured). To estimate all of the system states, a common approach is to form and solve the following least-squares problem:

$$\begin{aligned} \underset{\mathbf{u}}{\text{Minimise}} \quad \overline{J^{(T)}} &= \frac{1}{2T}\int_0^T (\mathbf{u}-\mathbf{u}_m)^T Q(\mathbf{u}-\mathbf{u}_m)\,dt \\ \text{subject to} \quad \mathbf{r}_{\mathbf{u}} &= \frac{d\mathbf{u}}{dt} - \mathbf{f}(\mathbf{u},t) = 0 \end{aligned} \tag{5.1}$$

which minimises the mismatch between the estimated states $\mathbf{u}(t)$ and the measurements $\mathbf{u}_m(t)$, subject to the non-linear ODE constraint. The matrix Q is $N \times N$ diagonal with

$$\begin{aligned} &Q_{i,i} = 1 \text{ if the } i^{th} \text{ state measurement is available,} \\ &Q_{i,i} = 0 \text{ if not available} \end{aligned} \tag{5.2}$$

The above minimisation statement (5.1) assumes that some states can be measured directly. If instead a linear combination of states $\mathbf{y}(t) = C\mathbf{u}(t)$ is measured (where C is the output matrix), the objective in (5.1) can be written as

$$\overline{J^{(T)}} = \frac{1}{2T}\int_0^T (C\mathbf{u}(t)-\mathbf{y}_m)^T (C\mathbf{u}(t)-\mathbf{y}_m)\,dt \tag{5.3}$$

where $\mathbf{y}_m$ is a vector of length M (number of measurements). It is shown in Appendix (C) that the resulting optimisation problem has a similar structure to that of (5.1). Data assimilation based on solving (5.1) or equivalent forms, has been carried out previously in [97, 12], where the aim was to find the initial state estimation that minimises the objective in (5.1) or (5.3). The current approach finds the full state estimation (i.e. at all time steps) that minimises the objective. This is to avoid the exponential error growth of guessing the initial condition.

It can be shown by using calculus of variations, that the optimal state estimation $\mathbf{u}(t)$ must satisfy the following optimality system (refer to Appendix (C) for the derivation):

$$\frac{d\mathbf{u}}{dt} - \mathbf{f}(\mathbf{u}, t) = 0 \tag{5.4a}$$

$$\frac{d\boldsymbol{\lambda}}{dt} + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\mathbf{u}}^{T} \boldsymbol{\lambda} + Q(\mathbf{u} - \mathbf{u}_m) = 0 \tag{5.4b}$$

$$\boldsymbol{\lambda}(0) = \boldsymbol{\lambda}(T) = 0 \tag{5.4c}$$

where $\boldsymbol{\lambda}(t)$ is the vector of adjoint variables. For chaotic systems, $\boldsymbol{\lambda}(t)$ should remain bounded in time because (5.4b) is constrained at both ends of the time window by the boundary conditions (5.4c). An iterative method to solve the system (5.4) must be used, since (5.4a) is non-linear. The Newton-Raphson method is well suited for this root-finding problem. More specifically, $\mathbf{u}(t)$ and $\boldsymbol{\lambda}(t)$ are updated as follows:

$$\begin{bmatrix} \mathbf{u}(t) \\ \boldsymbol{\lambda}(t) \end{bmatrix}_{(j+1)} = \begin{bmatrix} \mathbf{u}(t) \\ \boldsymbol{\lambda}(t) \end{bmatrix}_{(j)} + \begin{bmatrix} \delta\mathbf{u}(t) \\ \delta\boldsymbol{\lambda}(t) \end{bmatrix}_{(j)} \tag{5.5}$$

between iterations j and $j+1$. The update $\begin{bmatrix} \delta\mathbf{u}^T(t) & \delta\boldsymbol{\lambda}^T(t) \end{bmatrix}_{(j)}^T$ is computed by substituting (5.5) into (5.4), and linearising around $\begin{bmatrix} \mathbf{u}^T(t) & \boldsymbol{\lambda}^T(t) \end{bmatrix}_{(j)}$ to give the two-point boundary value problem (BVP):

$$\frac{d(\delta\mathbf{u})}{dt} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\mathbf{u}_{(j)}} \delta\mathbf{u}(t) - \mathbf{r}_{\mathbf{u}}(t) \tag{5.6a}$$

$$\frac{d(\delta\boldsymbol{\lambda})}{dt} = -Q\delta\mathbf{u}(t) - \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\mathbf{u}_{(j)}}^{T} \delta\boldsymbol{\lambda}(t) - \mathbf{r}_{\boldsymbol{\lambda}}(t) \tag{5.6b}$$

$$\delta\boldsymbol{\lambda}(0) = -\boldsymbol{\lambda}_{(j)}(0) \tag{5.6c}$$

$$\delta\boldsymbol{\lambda}(T) = -\boldsymbol{\lambda}_{(j)}(T) \tag{5.6d}$$

The boundary conditions (5.6 c,d) derive from (5.4c), and $\mathbf{r_u}(t)$ and $\mathbf{r}_{\boldsymbol{\lambda}}(t)$ are the residuals,

$$\begin{aligned} \mathbf{r_u}(t) &= \frac{d\mathbf{u}}{dt}_{(j)} - \mathbf{f}(\mathbf{u}_{(j)}) \\ \mathbf{r}_{\boldsymbol{\lambda}}(t) &= \frac{d\boldsymbol{\lambda}}{dt}_{(j)} + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\mathbf{u}_{(j)}}^{T} \boldsymbol{\lambda}_{(j)} + Q(\mathbf{u}_{(j)} - \mathbf{u}_m) \end{aligned} \tag{5.7}$$

The procedure to compute $\mathbf{u}(t)$ is straightforward, and the algorithm is presented below.

Algorithm 6: Data Assimilation

---

1. Set $j = 0$. Assume an initial distribution $\mathbf{u}(t)_{(0)} = \mathbf{u}_m(t)$ (i.e. the initial estimate uses full information of the measured states, while the other states are set to zero). Set $\boldsymbol{\lambda}_{(0)} = 0$.
2. Compute the residuals (5.7).
3. Call **Newton-Raphson Solver**[1] to compute $\begin{bmatrix}\delta\mathbf{u}^T(t) & \delta\boldsymbol{\lambda}^T(t)\end{bmatrix}^T$ by solving the two-point BVP (5.6).
4. Update the state and adjoint variables according to (5.5).
5. Set $j = j + 1$ and return to step 2, or break if $\mathbf{r_u}(t)$ and $\mathbf{r}_{\boldsymbol{\lambda}}(t)$ drop below a prescribed tolerance and output $\mathbf{u}_{(j)}(t)$ as the optimal estimation.

5.3 Multiple Shooting Method to Solve the Two-point BVP (5.6)

The multiple shooting method will be applied to solve the BVP (5.6). This method keeps storage costs low and relies on matrix-vector products only. A shooting method similar to that employed in Section (2.4) is derived below. Recall the tangent

[1]To be introduced in Section (5.3)

and adjoint state transition matrices that satisfy

$$\frac{d\phi_{\tau,t}}{dt} = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\bigg|_t \right) \phi_{\tau,t} \qquad\qquad \frac{d\phi^T_{\tau,t}}{d\tau} = -\left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\bigg|_\tau \right)^T \phi_{\tau,t} \tag{5.8}$$

The estimation time window is again split into P equal length segments. The analytical solutions to (5.6a) and (5.6b) within a given segment ($t_i \leq t \leq t_{i+1}$) are given respectively by

$$\delta\mathbf{u}(t) = \left(\phi_{t_i,t}\right)\delta\mathbf{u}_i - \int_{t_i}^{t} \left(\phi_{\tau,t}\right)\mathbf{r}_{\mathbf{u}}(\tau)\; d\tau \tag{5.9a}$$

$$\delta\boldsymbol{\lambda}(t) = \left(\phi^T_{t_{i+1},t}\right)\delta\boldsymbol{\lambda}_{i+1} - \int_{t_{i+1}}^{t} \left(\phi^T_{\tau,t}\right)\left(Q\delta\mathbf{u}(\tau) + \mathbf{r}_{\boldsymbol{\lambda}}(\tau)\right)\; d\tau \tag{5.9b}$$

The first terms on the right-hand-side of (5.9 a,b) are the zero-input response terms, while the second terms on the right-hand-side of (5.9 a,b) are the zero-state response terms.

The aim is to use the analytical solution form (5.9) to derive a coupled system of equations for $\delta\mathbf{u}_i$ and $\delta\boldsymbol{\lambda}_i$. Once $\delta\mathbf{u}_i$ and $\delta\boldsymbol{\lambda}_i$ are computed, they are used as initial conditions to integrate the coupled equations (5.6 a,b) in each segment to obtain the continuous time updates $\delta\mathbf{u}(t)$ and $\delta\boldsymbol{\lambda}(t)$. This concept is illustrated in Figure (5.1).

Equation (5.9a) is applied to all segments to find $\delta\mathbf{u}_{i+1}$ ($i = 0, 1, \ldots, P-1$),

$$\delta\mathbf{u}_{i+1} = \phi_{i+1}\delta\mathbf{u}_i + \mathbf{b}_{\delta\mathbf{u},i+1} \tag{5.10}$$

where $\phi_{i+1} = \phi_{t_i,t_{i+1}}$ and $\mathbf{b}_{\delta\mathbf{u},i+1} = -\int_{t_i}^{t_{i+1}} \left(\phi_{\tau,t_{i+1}}\right)\mathbf{r}_{\mathbf{u}}(\tau)\, d\tau$. Equation (5.10) is a set of P equations for $P+1$ unknowns ($\delta\mathbf{u}_0, \delta\mathbf{u}_1, \ldots \delta\mathbf{u}_P$). The adjoint equation (5.9b) is slightly more complicated due to the coupling term $-\int_{t_{i+1}}^{t} \left(\phi^T_{\tau,t}\right) Q\delta\mathbf{u}(\tau)\, d\tau$. It is assumed that if $\Delta T = t_{i+1} - t_i$ is sufficiently small, then the quantity $\left(\phi^T_{\tau,t}\right) Q\delta\mathbf{u}(\tau)$

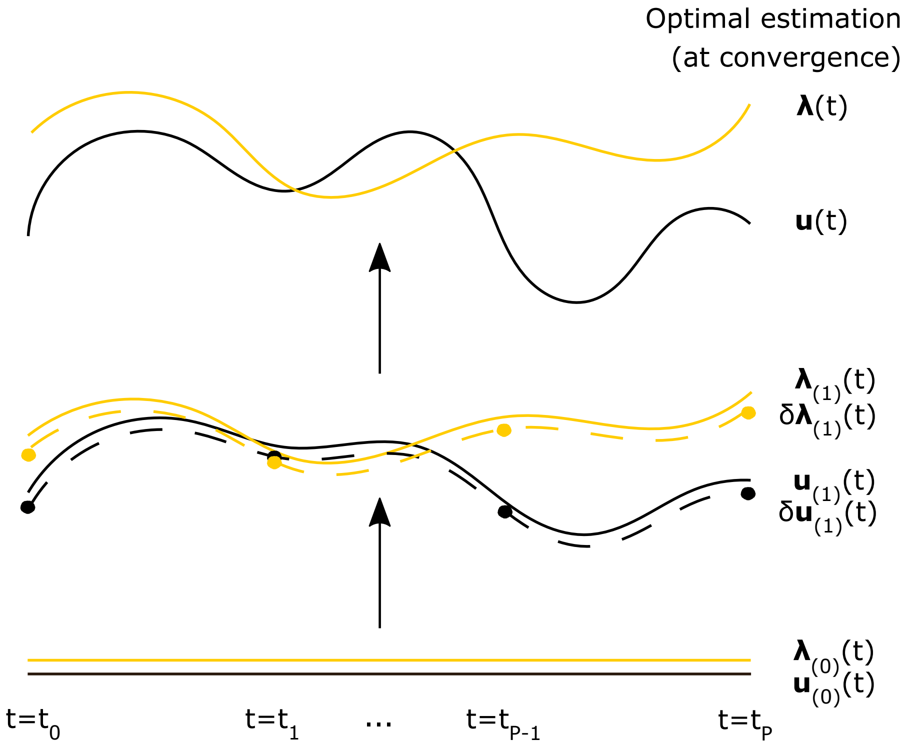


Figure 5.1: An illustration of the multiple shooting method used to update $\mathbf{u}_{(j)}(t)$ and $\boldsymbol{\lambda}_{(j)}(t)$. The discrete-time updates $\delta\mathbf{u}_i$ and $\delta\boldsymbol{\lambda}_i$ (shown as dots) are computed using equations (5.18) and (5.19), respectively. The solutions $\delta\mathbf{u}_i$ and $\delta\boldsymbol{\lambda}_i$ are then used as initial conditions for the integration of (5.6) in $t_i \le t \le t_{i+1}$ to obtain $\delta\mathbf{u}(t)$ and $\delta\boldsymbol{\lambda}(t)$ over the i^{th} segment.

is constant in the window $[t_{i+1}, t_i]$, and the following approximation can be made:

$$-\int_{t_{i+1}}^{t_i} \left(\phi^T_{\tau,t_i}\right) Q\delta\mathbf{u}(\tau)\,d\tau \approx \Delta T \left(\phi^T_{t_{i+1},t_i}\right) Q\delta\mathbf{u}_i \approx \Delta T Q\delta\mathbf{u}_i \tag{5.11}$$

Since $\phi^T_{t,t} = I$, then for small ΔT, $\phi_{t_{i+1},t_i} \approx I$. This is a crude approximation that only holds if $\Delta T \to 0$, meaning that ΔT should be small for the iterations of Algorithm (6) to converge. Proceeding in a similar fashion as in equation (5.10), the following discrete adjoint equations are derived:

$$\delta\boldsymbol{\lambda}_0 \approx \phi^T_1 \delta\boldsymbol{\lambda}_1 + \Delta T Q\delta\mathbf{u}_0 + \mathbf{b}_{\delta\boldsymbol{\lambda},0} \approx -\boldsymbol{\lambda}^{(j)}_0 \tag{5.12a}$$

$$\delta\boldsymbol{\lambda}_i \approx \phi^T_{i+1} \delta\boldsymbol{\lambda}_{i+1} + \Delta T Q\delta\mathbf{u}_i + \mathbf{b}_{\delta\boldsymbol{\lambda},i} \qquad (i = 1, 2, \ldots, P-1) \tag{5.12b}$$

$$\delta\boldsymbol{\lambda}_P = -\boldsymbol{\lambda}^{(j)}_P \tag{5.12c}$$

where $\mathbf{b}_{\delta\boldsymbol{\lambda},i} = -\int_{t_{i+1}}^{t_i} \left(\phi_{\tau,t_i}^T\right) \mathbf{r}_{\boldsymbol{\lambda}}(\tau)\, d\tau$. The boundary conditions (5.6c) and (5.6d) are imposed in (5.12a) and (5.12c), respectively.

Equations (5.10) and (5.12) define a coupled system of equations that can be solved for the discrete estimate and adjoint updates $\underline{\delta\mathbf{u}} = \begin{bmatrix} \delta\mathbf{u}_0^T & \delta\mathbf{u}_1^T & \dots & \delta\mathbf{u}_P^T \end{bmatrix}^T$ and $\underline{\delta\boldsymbol{\lambda}} = \begin{bmatrix} \delta\boldsymbol{\lambda}_1^T & \delta\boldsymbol{\lambda}_2^T & \dots & \delta\boldsymbol{\lambda}_P^T \end{bmatrix}^T$, respectively,

$$\begin{bmatrix} -G & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \underline{\delta\mathbf{u}} \\ \underline{\delta\boldsymbol{\lambda}} \end{bmatrix} = \begin{bmatrix} \underline{\mathbf{b}}_{\delta\boldsymbol{\lambda}} \\ \underline{\mathbf{b}}_{\delta\mathbf{u}} \end{bmatrix} \tag{5.13}$$

where

$$A = \begin{bmatrix} -\phi_1 & I & & & \\ & -\phi_2 & I & & \\ & & \ddots & \ddots & \\ & & & -\phi_P & I \end{bmatrix}, \quad \underline{\mathbf{b}}_{\delta\boldsymbol{\lambda}} = \begin{bmatrix} \mathbf{b}_{\delta\boldsymbol{\lambda},0} + \boldsymbol{\lambda}_0^{(i)} \\ \mathbf{b}_{\delta\boldsymbol{\lambda},1} \\ \vdots \\ \mathbf{b}_{\delta\boldsymbol{\lambda},P-1} \\ -\boldsymbol{\lambda}_P^{(i)} \end{bmatrix}, \quad \underline{\mathbf{b}}_{\delta\mathbf{u}} = \begin{bmatrix} \mathbf{b}_{\delta\mathbf{u},1} \\ \mathbf{b}_{\delta\mathbf{u},2} \\ \vdots \\ \mathbf{b}_{\delta\mathbf{u},P} \end{bmatrix} \tag{5.14}$$

and

$$G = diag\left(\Delta T Q, \dots, \Delta T Q, 0\right) \tag{5.15}$$

The matrix in (5.13) is similar to that of the MSS KKT matrix (2.23). Both are symmetric, indefinite, size $N(2P+1)$ square matrices with identical structure, but different blocks. Note also that no projection operators (2.18) are necessary.

To solve (5.13) efficiently using the preconditioner introduced in Section (3.4), the Schur complement needs to be formed, which requires inversion of the (1,1) block, G. However, this matrix is a diagonal that contains zero elements (also recall that Q (5.2) is not invertible). The review paper of Benzi et al. [50] highlights methods applicable for solving saddle point matrices (like equation 5.13) with singular top-left

(1,1) blocks. One approach [99] transforms (5.13) into an identical system with an invertible (1,1) block, for which a Schur complement may be formed. The resulting Schur complement is however expensive to form and challenging to precondition.

In order to invert the (1,1) block and form the Schur complement, G is regularised and replaced by

$$\tilde{G} = diag\left(\check{Q}, \ldots, \check{Q}, \epsilon I\right) \tag{5.16}$$

where $\check{Q} = \Delta T \tilde{Q}$ and

$$\begin{aligned} &\tilde{Q}_{i,i} = 1 \text{ if the } i^{th} \text{ state measurement is available,} \\ &\tilde{Q}_{i,i} = a \text{ if not available} \end{aligned} \tag{5.17}$$

The matrix $\tilde{Q}$ replaces the zero diagonal elements of Q (5.2) with a small positive number $0 < a \ll 1$, making $\tilde{G}$ invertible. The effect of replacing $Q_{i,i} = 0$ with $\tilde{Q}_{i,i} = a$ can be understood by inspecting the minimisation objective in (5.1). The weighting $Q_{i,i} = 0$ is required to ensure that unmeasured states are not minimised. Replacing $Q_{i,i} = 0$ with $\tilde{Q}_{i,i} = a$ means that very small weights are placed on the unmeasured states at the discrete points $(i = 0, 1, \ldots, P-1)$.

The parameter $0 < \epsilon \ll 1$ relaxes the boundary condition at $t = T$ (5.6c), i.e. the boundary condition becomes $-\epsilon I \delta \mathbf{u}_P + \delta \boldsymbol{\lambda}_P = -\boldsymbol{\lambda}_P^{(i)}$. Clearly a and ϵ should be as small as possible ($0 < a \ll 1$ and $0 < \epsilon \ll 1$). This is however challenging, as the Schur complement becomes more ill-conditioned as $a \to 0$ and $\epsilon \to 0$ (this will be discussed in Section 5.4).

The approximate Schur complement of (5.13) is

$$\left(A\tilde{G}^{-1}A^T\right)\underline{\delta\boldsymbol{\lambda}} = A\tilde{G}^{-1}\underline{\mathbf{b}}_{\delta\boldsymbol{\lambda}} + \underline{\mathbf{b}}_{\delta\mathbf{u}} \tag{5.18}$$

which is solved iteratively for $\underline{\delta\boldsymbol{\lambda}}$ and then substituted into

$$\underline{\delta\mathbf{u}} = \tilde{G}^{-1}\left(A^T\underline{\delta\boldsymbol{\lambda}} - \underline{\mathbf{b}}_{\delta\boldsymbol{\lambda}}\right) \tag{5.19}$$

Finally, (5.6) is integrated forward in time in all segments $i = 1, 2, \ldots, P$ with the initial conditions $\delta\mathbf{u}_{i-1}$ and $\delta\boldsymbol{\lambda}_{i-1}$ to obtain $\delta\mathbf{u}(t)$ and $\delta\boldsymbol{\lambda}(t)$. The matrix of (5.18) has similar spectral properties as the MSS matrix defined in (2.24). This also needs to be preconditioned for fast convergence. This will be the topic of the next section.

5.3.1 Preconditioning the Schur complement

The preconditioning approach of the system (5.18) is considered in this section and is implemented in the following section. The Schur complement matrix $S = A\tilde{G}^{-1}A^T$ of (5.18) is

$$S = \begin{bmatrix} \phi_1\check{Q}^{-1}\phi_1^T + \check{Q}^{-1} & -\check{Q}^{-1}\phi_2^T & & \\ -\phi_2\check{Q}^{-1} & \phi_2\check{Q}^{-1}\phi_2^T + \check{Q}^{-1} & -\check{Q}^{-1}\phi_3^T & \\ & \ddots & \ddots & \ddots \\ & & -\phi_P\check{Q}^{-1} & \phi_P\check{Q}^{-1}\phi_P^T + (\epsilon I)^{-1} \end{bmatrix} \tag{5.20}$$

The matrix S is symmetric, block tri-diagonal and positive definite. The matrix structure is identical to that of the MSS Schur complement matrix (2.24), however, the matrix blocks are different. The block diagonal preconditioner (3.14) derived in Section (3.4) can be used with minor changes. The preconditioner M_{BD} used here approximates the inverse of the diagonal of S, i.e.,

$$\mathrm{M}_{\mathrm{BD}} \approx \tilde{S}^{-1} = \begin{bmatrix} \left(\varphi_1\varphi_1^T\right)^{-1} & & & \\ & \left(\varphi_2\varphi_2^T\right)^{-1} & & \\ & & \ddots & \\ & & & \left(\varphi_P\varphi_P^T\right)^{-1} \end{bmatrix} \tag{5.21}$$

where for $i = 1, 2, \ldots, P-1$,

$$\varphi_i\varphi_i^T = \begin{bmatrix} \phi_i\check{Q}^{-\frac{1}{2}} & \check{Q}^{-\frac{1}{2}} \end{bmatrix} \begin{bmatrix} \check{Q}^{-\frac{1}{2}}\phi_i^T \\ \check{Q}^{-\frac{1}{2}} \end{bmatrix} = \phi_i\check{Q}^{-1}\phi_i^T + \check{Q}^{-1} \tag{5.22}$$

and for $i = P$,

$$\varphi_P\varphi_P^T = \begin{bmatrix} \phi_P\check{Q}^{-\frac{1}{2}} & (\epsilon I)^{-\frac{1}{2}} \end{bmatrix} \begin{bmatrix} \check{Q}^{-\frac{1}{2}}\phi_P^T \\ (\epsilon I)^{-\frac{1}{2}} \end{bmatrix} = \phi_P\check{Q}^{-1}\phi_P^T + (\epsilon I)^{-1} \tag{5.23}$$

Note that (5.21) is identical in structure to (3.13). The matrix φ_i is $N \times 2N$ which can be written as

$$\varphi_i = U_i\Sigma_i V_i^T \tag{5.24}$$

where U is a $N \times N$ matrix containing the N left singular vectors, Σ is a $N \times 2N$ quasi-diagonal matrix containing the N singular values $\sigma(\varphi)$ and V is a $2N \times 2N$ matrix containing the $2N$ right singular vectors. The N singular values $\sigma(\varphi)$ are contained in the $N \times N$ diagonal sub-matrix (the last N columns consist of zeros, which are ignored).

The preconditioner M_{BD} (3.14) based on partial SVD is repeated here:

$$\mathrm{M_{BD}}_{(l)}^{(q)} = diag(\mathrm{M}_{(l),1}^{(q)}, \mathrm{M}_{(l),2}^{(q)}, ..., \mathrm{M}_{(l),P}^{(q)}) \tag{5.25a}$$

$$\mathrm{M}_{(l),i}^{(q)} = U_{1,i}\Sigma_{1,i}^{-2}U_{1,i}^T + (I - U_{1,i}U_{1,i}^T) \tag{5.25b}$$

Now $\Sigma_{1,i}$ is the diagonal matrix containing the l leading singular values $\sigma(\varphi)$ with corresponding left singular vectors $U_{1,i}$. In summary, the same preconditioner is used, but the singular values and vectors of the matrix φ_i replace those of the matrix Φ_i.

$\mathrm{M_{BD}}_{(l)}^{(q)}$ is used as a left preconditioner for the system (5.18) with regularisation applied:

$$\left(\gamma I + \mathrm{M_{BD}}_{(l)}^{(q)} S\right) \underline{\boldsymbol{\delta\lambda}} = \mathrm{M_{BD}}_{(l)}^{(q)} \left(A\tilde{G}^{-1}\underline{\mathbf{b}}_{\delta\boldsymbol{\lambda}} + \underline{\mathbf{b}}_{\delta\mathbf{u}}\right) \tag{5.26}$$

Appendix (D) shows how to to compute the MATVEC product $\left(\gamma I + \mathrm{M_{BD}}_{(l)}^{(q)} S\right) \underline{\mathbf{z}}$ for an arbitrary vector $\underline{\mathbf{z}}$. The algorithm to compute the Newton-Raphson update (5.6), i.e. step 3 of Algorithm (6), is summarised below.

Algorithm 7: Newton-Raphson Solver

Inputs: T, P, a, ϵ, l, q, γ. Outputs: $\delta\mathbf{u}(t)$, $\delta\boldsymbol{\lambda}(t)$

1. Form the vector $\left(A\tilde{G}^{-1}\underline{\mathbf{b}}_{\delta\boldsymbol{\lambda}} + \underline{\mathbf{b}}_{\delta\mathbf{u}}\right)$ (refer to Appendix D for more details).
2. Compute the preconditioner $\mathrm{M_{BD}}_{(l)}^{(q)}$ using (5.25).
3. Form the RHS of (5.26).
4. Solve (5.26) for $\underline{\boldsymbol{\delta\lambda}}$ using GMRES.
5. Compute $\underline{\boldsymbol{\delta}\mathbf{u}}$ using (5.19).
6. Integrate (5.6) forward in time in segments $i = 1, 2, \ldots, P$ with initial conditions $\delta\mathbf{u}_{i-1}$ and $\delta\boldsymbol{\lambda}_{i-1}$ to obtain $\delta\mathbf{u}(t)$ and $\delta\boldsymbol{\lambda}(t)$.

Algorithm (6) therefore has two iterative loops: the outer loop and the inner loop (the iterative solution of (5.26), i.e. step 4 of Algorithm 7).

5.4 Application to the Lorenz System

Algorithm (6) was applied to obtain optimum state estimations $\mathbf{u}(t) = \begin{bmatrix} x & y & z \end{bmatrix}^T$ for the Lorenz system, which is repeated below for convenience,

$$\frac{dx}{dt} = \sigma(y - x) \qquad \frac{dy}{dt} = x(\rho - z) - y \qquad \frac{dz}{dt} = xy - \beta z \tag{5.27}$$

The standard Lorenz parameters ($\sigma = 10$, $\beta = {}^8\!/_3$ and $\rho = 28$) were used throughout this chapter. The multiple shooting method introduced in the previous section and summarised in Algorithm (7) was used to compute the Newton-Raphson update (5.6). Initially, the two variables $y(t)$ and $z(t)$ were measured, while $x(t)$ was estimated. Therefore the matrix $Q = diag(0, 1, 1)$, and $\tilde{Q} = diag(a, 1, 1)$. In Section (5.4.3), a more challenging case where only one variable is measured and the other two are estimated, is also considered.

Equation set (5.27) was integrated in $[0, T]$ with an initial condition $\mathbf{u}_0$ on the attractor, to obtain a trajectory $\mathbf{u}_{true}(t) = \begin{bmatrix} x_{true} & y_{true} & z_{true} \end{bmatrix}^T$. The measurement vector $\mathbf{u}_m(t) = \begin{bmatrix} 0 & y_{true} & z_{true} \end{bmatrix}^T$ was defined and the aim was to recover $\mathbf{u}_{true}(t)$ from an initial state estimate $\mathbf{u}_{(0)}(t) = [0 \quad y_{true} \quad z_{true}]^T$ and initial adjoint $\boldsymbol{\lambda}_{(0)}(t) = [0 \quad 0 \quad 0]^T$. The algorithm was set to terminate when all time-averaged absolute residuals $\overline{|\mathbf{r}_\mathbf{u}(t)|}$ and $\overline{|\mathbf{r}_{\boldsymbol{\lambda}}(t)|}$ have dropped below 1×10^{-3}. A general test case is presented first, followed by a more detailed analysis of the effect of varying different parameters.

Figure (5.2) shows the convergence of the three components of $\overline{|\mathbf{r}_\mathbf{u}(t)|}$ and $\overline{|\mathbf{r}_{\boldsymbol{\lambda}}(t)|}$ for a trajectory $T = 50$. The relaxation parameters used were $a = 1 \times 10^{-5}$ and $\epsilon = 1 \times 10^{-3}$, while the solver (5.26) used $\Delta T = 0.2$, $\gamma = 1 \times 10^{-5}$ and $l = 3$. Also superimposed is $\overline{J^{(T)}}$, to monitor the departure of $\mathbf{u}_{(j)}$ from $\mathbf{u}_m$. Both $\overline{J^{(T)}}$ and $\overline{|\mathbf{r}_{\boldsymbol{\lambda}}(t)|}$ were initially zero (since $\mathbf{u}_{(0)}(t) = \mathbf{u}_m(t)$ and $\boldsymbol{\lambda}_{(0)}(t) = 0$). As clearly shown, the outer iterations converge very fast and $\overline{|\mathbf{r}_\mathbf{u}(t)|}$ drops by at least 3 orders

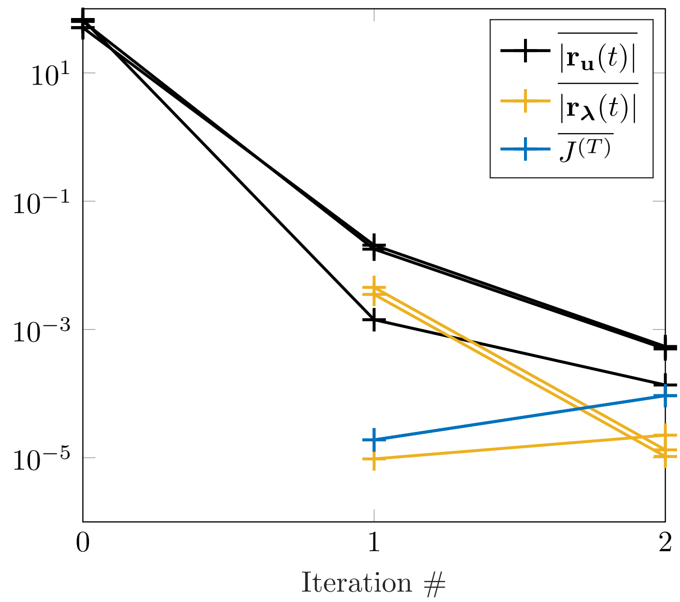


Figure 5.2: Time-averaged residual convergence for a trajectory $T = 50$. The solver parameters used were $\Delta T = 0.2$, $\gamma = 1 \times 10^{-5}$ and $l = 3$, while the relaxation parameters used were $a = 1 \times 10^{-5}$ and $\epsilon = 1 \times 10^{-3}$.

of magnitude by the first iteration. By the second iteration, all residuals are below 1×10^{-3} and $\overline{J^{(T)}} \approx 1 \times 10^{-4}$. The convergence rate, although fast, stalls. This is not typical of the Newton-Raphson method which guarantees quadratic convergence. However, since an approximate equation (5.26) is used to compute the discrete-time updates $\delta\boldsymbol{\lambda}_i$ and $\delta\mathbf{u}_i$, convergence to machine zero cannot be achieved. The initial and converged residuals as a function of t are shown in Figure (5.3). The initial

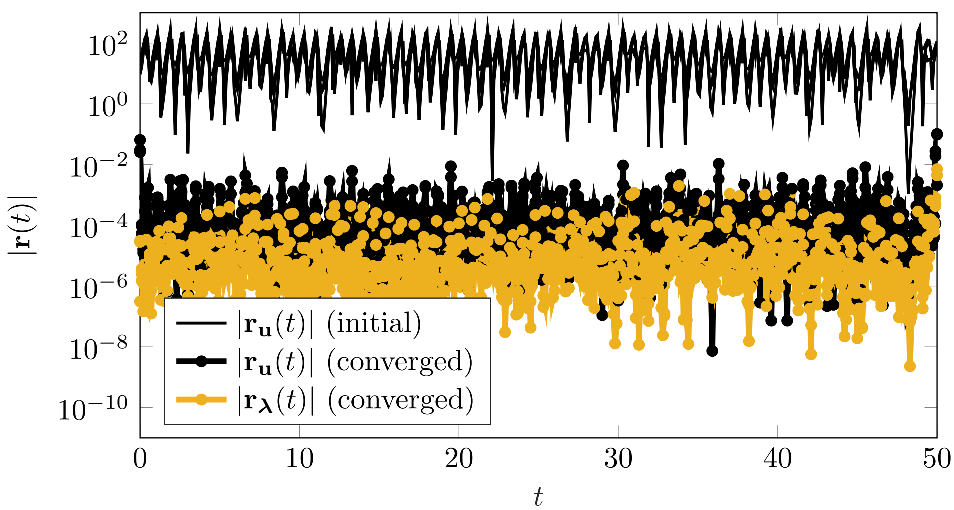


Figure 5.3: Absolute values of the initial and converged (iteration #2) residuals.

adjoint residual $\mathbf{r}_{\boldsymbol{\lambda}(0)}(t) = 0$, so it is not shown. It is clear that $\mathbf{r}_{\mathbf{u}}(t)$ drops by several orders of magnitude along the entire assimilation time window.

Figure (5.4) shows the estimated variable x, along with the true variable x_{true}, obtained by integrating (5.27). The estimation is indistinguishable from the true trajectory, demonstrating the effectiveness and accuracy of Algorithm (6).

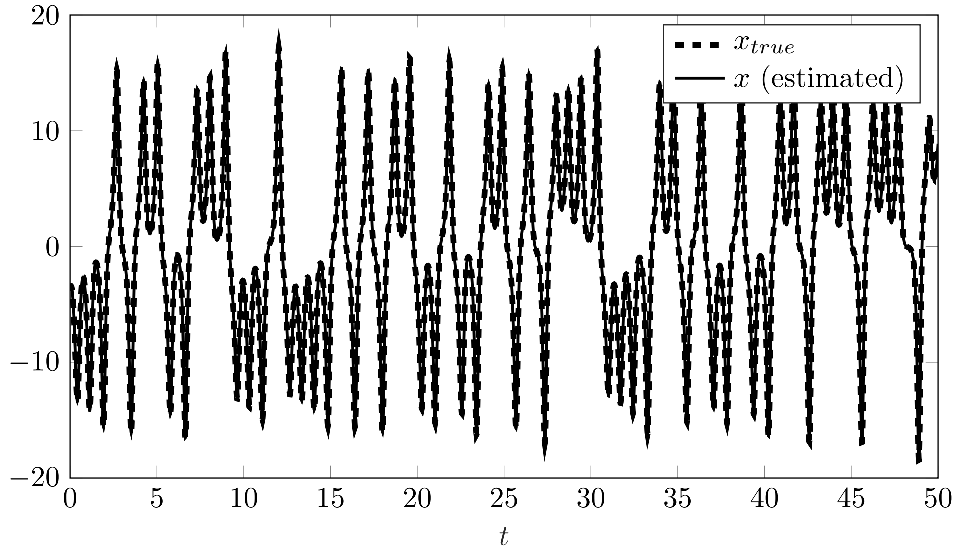


Figure 5.4: A comparison of the estimated variable x with the true solution x_{true}.

5.4.1 Analysis of the effect of preconditioning

The effect of preconditioning on the eigenvalue spectrum and the convergence rate is examined here. As will be shown in the following subsection, the parameter a was found to have an important effect on the convergence rate of the outer iterations and the estimation quality. It is therefore important to look at the properties of the preconditioned matrix for a range of a values.

The singular values $\sigma(\varphi_i)$ defined in (5.22) and (5.23), which are used to construct the preconditioner, are studied first. In Figure (5.5), $\sigma(\varphi_i)$ $(i = 1, 2, \ldots, P)$, ordered from largest to smallest, are plotted for different values of a. The vertical dashed lines separate $\sigma(\varphi_i)$ into three groups: those corresponding to $\lambda'_{1,i} > 0$ (left), $\lambda'_{2,i} = 0$

(centre) and $\lambda'_{3,i} < 0$ (right), where λ'_i is the finite-time Lyapunov exponent in each segment i. It can be seen that reducing a by an order of magnitude (for $a \leq 1\times10^{-1}$), increases most $\sigma(\varphi_i)$ in the left and centre groups by a factor of $\approx \sqrt{10}$. The smallest $\sigma(\varphi_i)$ (right group) remain mostly unaffected by changes to a.

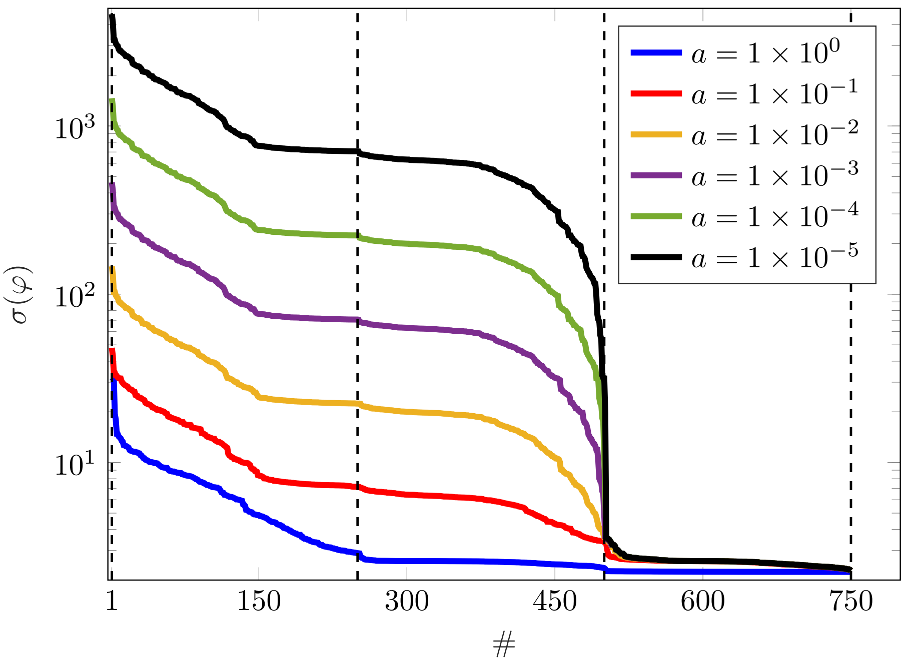
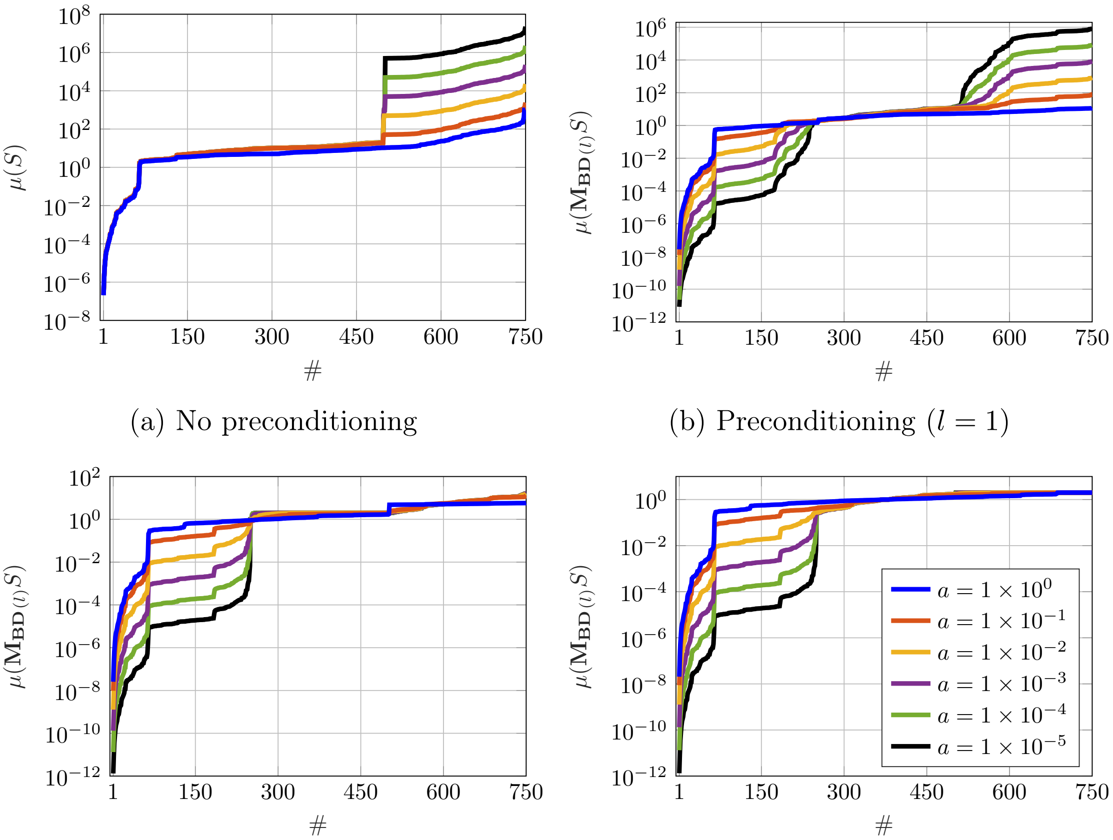


Figure 5.5: Singular values $\sigma(\varphi)$, plotted for different values of a. There are $P = 250$ segments and therefore a total of $250 \times 3 = 750$ singular values.

Figure (5.6) shows the eigenvalues of S (Panel a) and of $\mathrm{M}_{\mathrm{BD}(l)}S$ (preconditioned Schur complement) for $l = 1, 2, 3$ (Panels b-d), ordered from smallest to largest. No Tikhonov regularisation was applied ($\gamma = 0$). Without preconditioning applied (Panel a), the largest 250 eigenvalues (corresponding to the positive Lyapunov exponent) are widely spread, which leads to slow convergence of the GMRES solver for the internal iterations. It can be seen that $\mu_{max}(S)$ increases by an order of magnitude when a is reduced by the same order. Using $l = 1$ (Panel b), $\mu_{max}(\mathrm{M}_{\mathrm{BD}(1)}S)$ is reduced by approximately two orders of magnitude, but the clustering of the largest eigenvalues remains poor for the smaller values of a. Using $l = 2$ (Panel c) significantly improves the clustering of the largest eigenvalues, while using $l = 3$ (Panel d) ensures that $\mu_{max}(\mathrm{M}_{\mathrm{BD}(3)}S) \approx 2$ for any value of a.



(a) No preconditioning

(b) Preconditioning ($l = 1$)

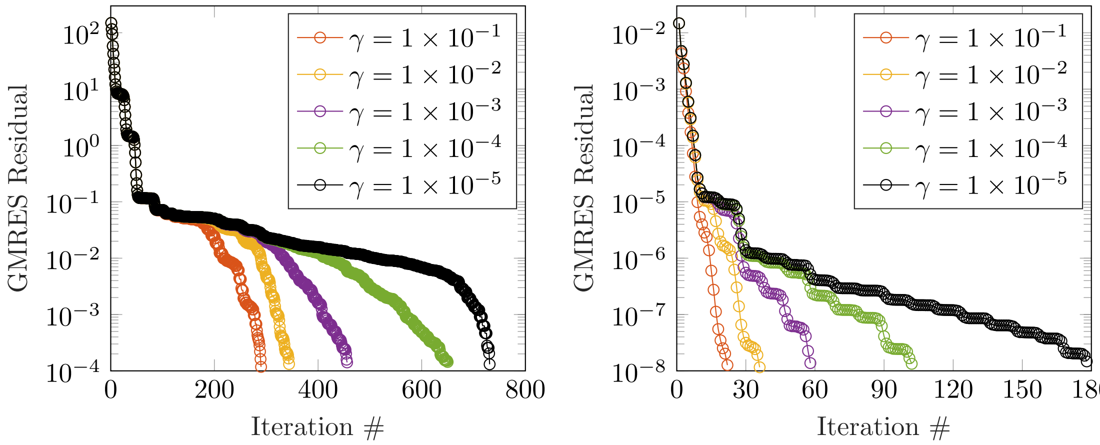
(c) Preconditioning ($l = 2$)

(d) Preconditioning ($l = 3$)

Figure 5.6: Eigenvalue spectra of the Schur complement S and the preconditioned Schur complement $\mathrm{M}_{\mathrm{BD}(l)}S$. All spectra were obtained for $T = 50$, $\Delta T = 0.2$, $\gamma = 0$ and $\epsilon = 1 \times 10^{-3}$.

While the preconditioner has been shown to effectively cluster the largest eigenvalues, it is clear that $\mathrm{M}_{\mathrm{BD}(l)}S$ becomes more singular as a is reduced (notice the drop in the smallest 250 eigenvalues in Panels (b-d) with decreasing a). This is attributed to the loss of the block diagonal dominance of $\mathrm{M}_{\mathrm{BD}(l)}S$ with decreasing a, therefore making the approximation (5.21) less accurate.

It was found that the convergence of $\overline{|\mathbf{r}_{\mathbf{u}}(t)|}$ and $\overline{|\mathbf{r}_{\boldsymbol{\lambda}}(t)|}$ became increasingly sensitive to γ with decreasing a. In other words, the smaller the value of a, the smaller γ was required to ensure that $\overline{|\mathbf{r}_{\mathbf{u}}(t)|}$ and $\overline{|\mathbf{r}_{\boldsymbol{\lambda}}(t)|}$ have converged. Unfortunately, smaller γ leads to slower convergence of GMRES (with or without preconditioning). This can be clearly seen in Figure (5.7). Notice however that the initial convergence rate is independent of γ. For the initial residual to drop by two orders of magnitude (for any γ value), 32 iterations without preconditioning vs. 6 preconditioner iterations were required. The fast initial preconditioned convergence rate can be exploited by relaxing the GMRES solver tolerance. The next subsection considers in detail the effect of varying a on the estimation error and the outer loop convergence rate.



(a) No preconditioning (b) Preconditioning with $l = 3$

Figure 5.7: Effect of varying γ on the GMRES convergence rate, whilst fixing all other parameters: $T = 50$, $\Delta T = 0.2$, $l = 3$, $a = 1 \times 10^{-5}$ and $\epsilon = 1 \times 10^{-3}$. The residuals shown are for the first 'outer iteration'.

5.4.2 Effect of varying the relaxation parameters a and ϵ

The previous subsection discussed the use of the preconditioner to accelerate the GMRES convergence. In this subsection, the effect of the relaxation parameters a and ϵ on the outer loop convergence rate and the estimation error is considered. The effect of varying a is discussed first.

Figure (5.8) shows the convergence of $\overline{|\mathbf{r_u}(t)|}$ and $\overline{|\mathbf{r_\lambda}(t)|}$ for different values of a. The same stopping criteria as before was used, i.e. to terminate when $\overline{|\mathbf{r_u}(t)|}$ and $\overline{|\mathbf{r_\lambda}(t)|}$ drop below 1×10^{-3}. Only γ was allowed to vary from one simulation to the other (to avoid divergence at small values of a). As expected, the convergence of the outer iterations $\overline{|\mathbf{r_u}(t)|}$ and $\overline{|\mathbf{r_\lambda}(t)|}$ is accelerated as a is reduced, because the solution to the original system (5.13) is approximated best as $a \to 0$. However, saturation occurs at $a = 1 \times 10^{-3}$ (convergence rate is very similar for $a = 1 \times 10^{-3}$ and $a = 1 \times 10^{-5}$). The value of $\overline{J^{(T)}}$ on the final iteration decreases with a, and this reflects directly on the absolute error of the estimation $|x - x_{true}|$, as shown in Figure (5.9). Clearly, smaller values of a result in more accurate estimations. It is also worth noting that the outer loop iterations diverged for values greater than $a \approx 2.5 \times 10^{-1}$.

Table (5.1) summarises the results shown in Figures (5.8) and (5.9). The fourth column is included to show the average number of GMRES iterations executed per outer iteration and the combined number (over all outer iterations). A GMRES

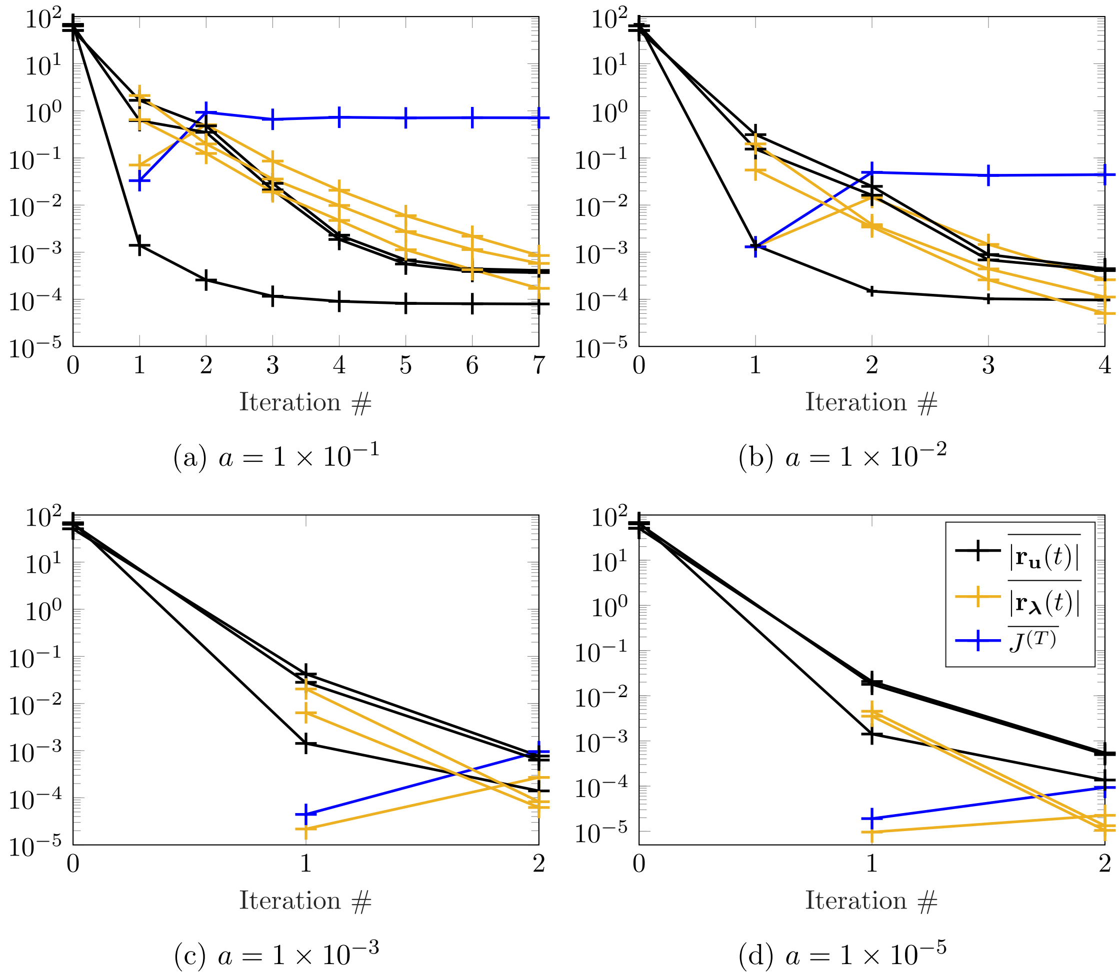


Figure 5.8: Effect of varying the relaxation parameter a on the residual convergence for a $T = 50$ trajectory. The parameters $\Delta T = 0.2$, $l = 3$, and $\epsilon = 1 \times 10^{-3}$ were used. The regularisation parameter γ was varied according to the second column of Table (5.1).

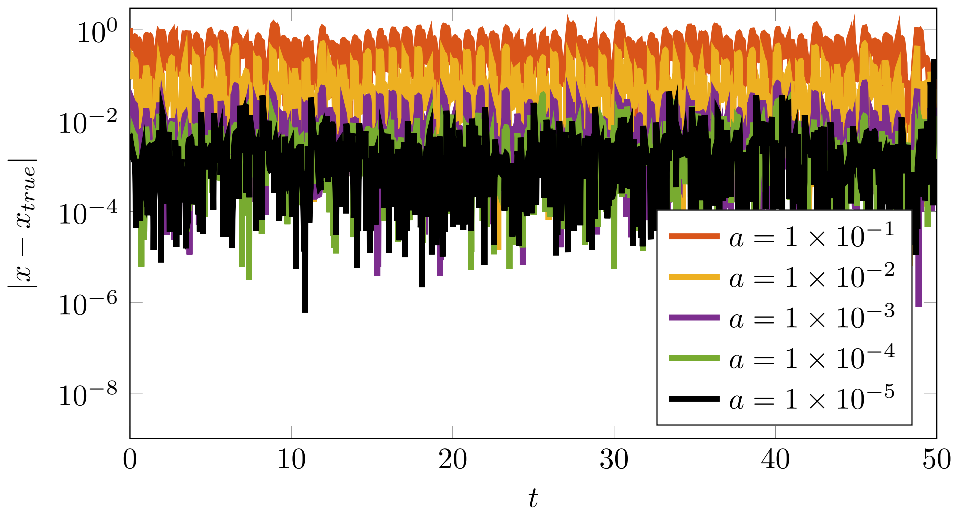


Figure 5.9: Effect of varying a on the estimation absolute error $|x - x_{true}|$. The errors were computed on the final (converged) outer iteration.

relative tolerance tol $= 1 \times 10^{-3}$ was prescribed (tol $= \|r_{initial}\|_2 / \|r_{final}\|_2$, where r is the residual of equation 5.26) for all a (except for $a = 1 \times 10^{-5}$, where tol $= 1 \times 10^{-4}$ was used). It was found that smaller tolerances lead to diminishing returns on the estimation error and the outer iteration convergence rate, justifying the aforementioned choice of tol, while keeping the GMRES convergence rate independent of γ (recall that the convergence rate is initially independent on γ, as shown in Figure 5.7b). The number of iterations till convergence for $a \geq 1 \times 10^{-4}$ is almost independent of a (it is higher for $a = 1 \times 10^{-5}$ because a tighter tolerance was used). The time-averaged absolute error $\overline{|x - x_{true}|}$ drops sharply with a, but stagnates at $a = 1 \times 10^{-4}$.

a	γ	# outer iterations	# GMRES (inner) iterations per outer it.	total	absolute error $\overline{|x - x_{true}|}$
1×10^{-1}	1×10^{-2}	7	22	154	3.96×10^{-1}
1×10^{-2}	1×10^{-2}	4	20	80	7.66×10^{-2}
1×10^{-3}	1×10^{-3}	2	23	46	8.30×10^{-3}
1×10^{-4}	1×10^{-4}	2	23	46	3.50×10^{-3}
1×10^{-5}	1×10^{-5}	2	31	62	3.40×10^{-3}

Table 5.1: Summary of the key performance indicators for different values of a. It was necessary to reduce γ with decreasing a to ensure convergence of the outer iterations. The GMRES solver used a relative tolerance tol $= 1 \times 10^{-3}$ (except for $a = 1 \times 10^{-5}$, where tol $= 1 \times 10^{-4}$ was used). The absolute error was computed on the final (outer) iteration.

The effect of varying ϵ is now considered. Recall that ϵ relaxes the boundary condition at $t = T$ (5.6c). As shown in Figure (5.10), ϵ mostly affects the error $|x(t) - x_{true}(t)|$ close to the end of the trajectory, while the effect is insignificant at earlier t. It was found that ϵ had a negligible effect on the convergence rate of $\overline{|\mathbf{r_u}(t)|}$ and $\overline{|\mathbf{r_\lambda}(t)|}$ (the outer iterations) for the values of ϵ shown in Figure (5.10).

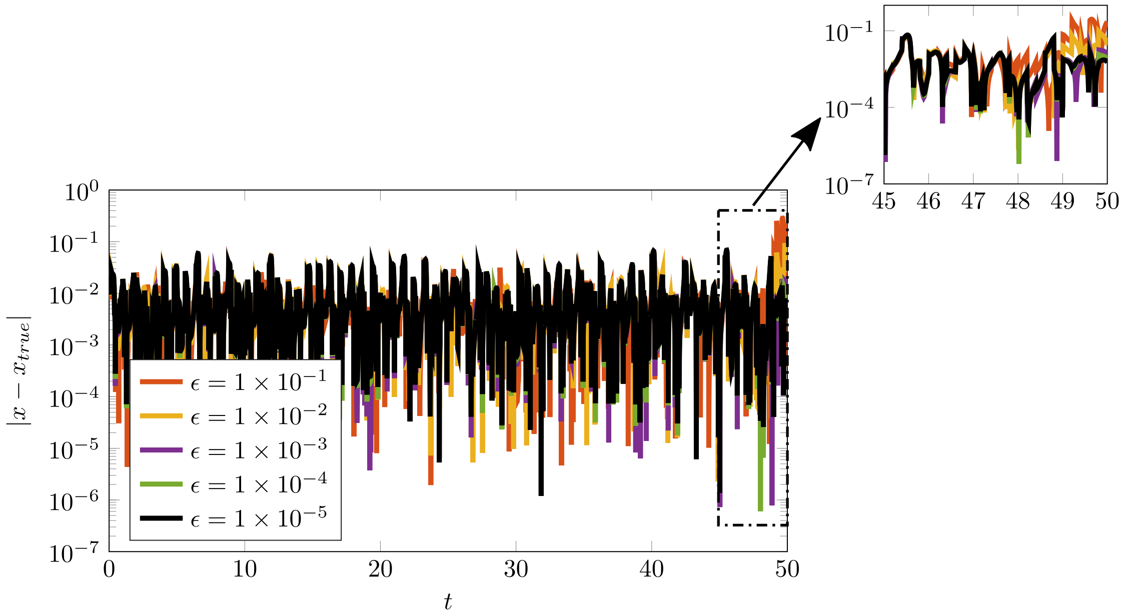


Figure 5.10: Effect of varying ϵ on the estimation absolute error $|x - x_{true}|$. The errors were computed on the final (converged) outer iteration. The parameters $a = 1 \times 10^{-3}$ and $\gamma = 1 \times 10^{-3}$ were used.

5.4.3 Estimation of two states

The results presented thus far assumed that a single state x_{true} was estimated. It is important to examine the response of Algorithm (6) when a smaller number of states are measured. In this subsection, it was assumed that only x_{true} was available, while y_{true} and z_{true} must be estimated. The measurement vector is therefore $\mathbf{u}_m(t) = [x_{true} \quad 0 \quad 0]^T$ and the aim was to estimate $\mathbf{u}_{true}(t)$, given $\mathbf{u}_m$. The matrix $Q = diag(1, 0, 0)$ and $\tilde{Q} = diag(1, a, a)$.

As shown in Figure (5.11), the convergence behaviour of $\overline{|\mathbf{r_u}(t)|}$ and $\overline{|\mathbf{r_\lambda}(t)|}$ (computed for $a = 1 \times 10^{-6}$) is similar to that for a single-state estimation (as shown in Figure (5.2) for $a = 1 \times 10^{-5}$). Only one additional iteration was required to satisfy the same outer loop convergence tolerance (1×10^{-3}).

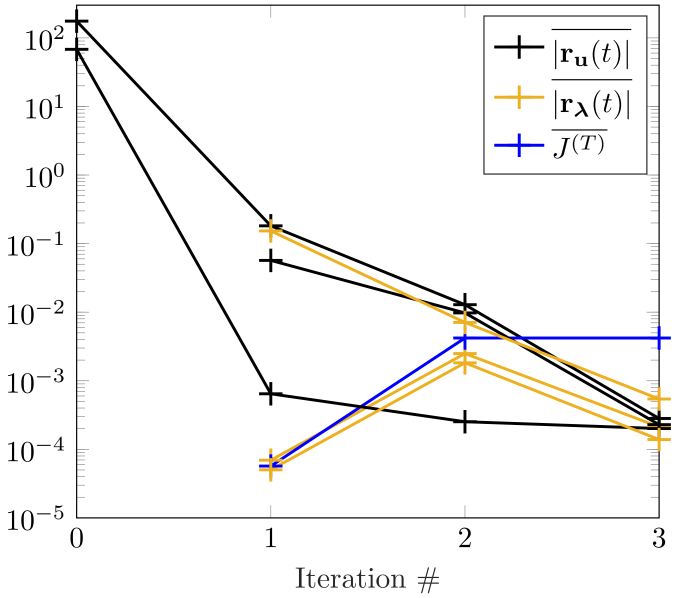


Figure 5.11: Time-averaged residual convergence when estimating the states y and z over $T = 50$. The solver parameters used were $\Delta T = 0.2$, $\gamma = 1 \times 10^{-7}$ and $l = 3$, and the relaxation parameters used were $a = 1 \times 10^{-6}$ and $\epsilon = 1 \times 10^{-6}$.

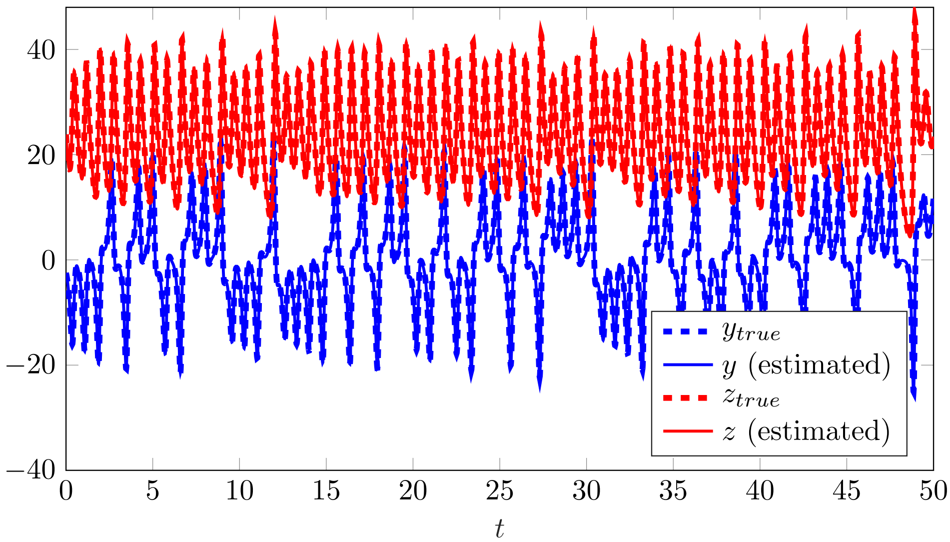


Figure 5.12: Comparison of the estimated variables x and y, with the true solutions x_{true} and y_{true}, respectively.

Figure (5.12) shows the comparison of the estimated states and their true values. The time-averaged absolute errors for this case were $\overline{|y - y_{true}|} = 9.74 \times 10^{-2}$ and

$\overline{|z - z_{true}|}$ =1.34×10^{-1}. Although these errors are larger than $\overline{|x - x_{true}|}$ =3.10×10^{-3} (computed for a single-state estimation with the same parameters), it is still possible to construct very accurate estimations from less measurement data, at least for the Lorenz system.

5.5 Summary

In this chapter, a multiple shooting method for the state reconstruction of chaotic systems was formulated. The method uses outer iterations to reduce the residuals (5.7) and inner iterations to solve the linear Schur complement matrix system (5.26) using GMRES. The preconditioner introduced in Section (3.4) was used (with minor changes) to accelerate the convergence rate of the Schur complement system. Results were presented for estimating Lorenz system states.

It was shown that the preconditioner deflated the maximum eigenvalue to $\mu_{max} \approx 2$ for all values of a (with the correct choice of the number of singular modes l). The preconditioner did, however, cause μ_{min} to drop with decreasing a. While it was still possible to keep the (inner loop) GMRES convergence rate independent of a (by increasing the GMRES solver tolerance), further work on the development of more accurate preconditioning is required.

Estimated states were computed with high accuracy. The outer loop convergence rate and the estimation accuracy were influenced mostly by the relaxation parameter a. The optimum value of a to use would likely be system dependent and also dependent on the number of estimated states. For the Lorenz System, the smallest estimation errors were obtained for $a = 1 \times 10^{-5}$ (one state estimation) and $a = 1 \times 10^{-6}$ (two-state estimation). Further reduction in a led to diminishing returns on the estimation error and the outer loop convergence rate.

The preconditioning of saddle point systems (analagous to equation 5.13) arising from the application of variational data assimilation to nonlinear systems, has only recently caught attention [98, 100, 101]. The current approach differs by considering the partial SVD of the state transition matrices (linear operators) to precondition the Schur complement system. For this approach to be competitive, an alternative preconditioner structure that bounds the minimum eigenvalue μ_{min} should be devised for future work.

Chapter 6

Conclusions and Future Work

6.1 Summary of the Main Contributions

The successful application of shadowing algorithms for the sensitivity analysis of chaotic systems has been demonstrated in the past. The LSS method is well conditioned because it relaxes the initial condition and subsequently solves a least-squares problem to find the trajectory for the perturbed parameter, which shadows the reference trajectory. The demanding memory requirements of LSS and the very long computational time make it virtually impossible to apply to large systems. The multiple shooting variant (MSS) has only moderate storage requirements, but still suffers from slow convergence due to the very large condition number of the system matrix. The main aim of this thesis was to reduce the computational cost of the shadowing algorithm MSS and to apply the developed tools for the feedback control and data assimilation of chaotic systems.

To accelerate the convergence rate of the MSS Schur complement system, a block diagonal preconditioner based on partial singular value decomposition was proposed. The preconditioner works by annihilating the fastest growing modes (corresponding to the largest positive Lyapunov exponents), while Tikhonov regularisation is used to regularise the smallest eigenvalues. This tight clustering of eigenvalues leads to very fast convergence. With an appropriate choice of the parameters l and γ, the convergence rate was found to be almost independent of the trajectory length and the #DOF for the 1D KSE. Similar convergence behaviour can be expected for larger systems if l and γ are chosen suitably. It was shown that the singular modes of the MSS matrix A corresponding to very small singular values ($0 < \sigma(A) \ll 1$) were responsible for a reproducible bias in the sensitivities of the KSE and the Lorenz system (for certain parameter values). Very small $\sigma(A)$ are present for non-uniformly hyperbolic systems due to tangencies of the Lyapunov vectors at some points along the attractor. By expressing the minimum norm solution of the MSS problem in terms of a sum involving the singular modes of A, it was found that the sensitivity bias can be almost eliminated with the removal of all modes with $\sigma(A) < 1$.

The adjoint version of preconditioned MSS was then coupled with a gradient descent algorithm to compute a feedback control matrix for the KSE. The control algorithm updates all elements of the feedback matrix with a single adjoint computation. To the best of the author's knowledge, this was the first attempt to use shadowing to compute a feedback controller. A stabilising controller was found with just one iteration of the algorithm, demonstrating its effectiveness. The response of the actuated system to the feedback controllers computed using PMSS and standard LQR were compared. The performance, measured in terms of how fast the actuated systems were stabilised, was used to compare both controllers. The LQR controller led to faster stabilisation of the actuated system compared to the PMSS controller. It is thought however that LQR theory is ideal for systems with passive energy conserving non-linearities (such as the KSE with zero Dirichlet and Neumann boundary conditions). The PMSS controller could outperform LQR for systems that are not non-linearly energy conserving, i.e. passive (this of course needs to be confirmed). Furthermore, the solution to the Riccati equation (whose cost scales with $O(N^3)$) is bypassed when applying PMSS control.

It was shown that the applicability of the preconditioner is not exclusive to MSS. With minor modifications, the preconditioner was used to speed up the convergence rate of the linear matrix system of a variational data assimilation algorithm. When applied to the Lorenz system, accurate estimations of two states were computed (assuming knowledge of only the third state). While the preconditioner effectively deflated the largest eigenvalues μ of the system close to $\mu = 1$, some eigenvalues approached zero. Nonetheless, the preconditioner was again able to cut down significantly on the number of iterations.

6.2 Future Work

- Further work is required to determine a suitable preconditioning parameter l. Knowledge of the number of positive Lyapunov exponents would be sufficient, but since this information is usually unavailable, the computed singular values for each segment should be used to guide the choice of l. An adaptive algorithm can be easily devised. The value of l can be made to vary between segments and can be equal to the number of $\sigma(\Phi_i) > 1$ within a given segment, i.e. approximately equal to the number of unstable modes. This can be achieved by either computing more modes than necessary and truncating the unnecessary ones (i.e. truncating those with $\sigma(\Phi_i) \leq 1$), or by computing a small initial number of modes and augmenting as necessary until all unstable modes (with $\sigma(\Phi_i) > 1$) have been computed.

- The parameter l is expected to increase as the number of positive Lyapunov exponents increases, and the later would be the limiting factor for applying MSS to turbulent flow systems at large Reynolds numbers. Methods that reduce the dependence of the computational cost on the number of positive

Lyapunov exponents would be extremely valuable, and more research should focus on this direction.

- The PMSS control algorithm should be applied to larger systems. For example, turbulent flow in a channel at low Reynolds number would be a good starting point. It would be useful to see how the attractor changes with the iteration number and if an optimum solution can indeed be found. Potential challenges include the loss of ergodicity with changes in the intermediate matrix values $K^{(i+1)}$.

- In practical applications, only noisy measurements at some locations, say $\mathbf{y}_j(t)$, are available. This is known as output feedback control; in this case the actuation must be expressed as $\mathbf{s}(t) = g(\mathbf{y}_j)$. Assuming g to be a linear function of present and past values of the measurements (to account for memory effects), one can write $\mathbf{s}(t) = K_0\mathbf{y}_j(t) + K_1\mathbf{y}_j(t - \delta t) + ... + K_m\mathbf{y}_j(t - m\delta t)$, where the matrices $K_0, K_1, .., K_m$ can be obtained by applying the PMSS Control algorithm presented in Chapter (4) (instead of neural networks as for example in [102]). The value of m is problem dependent and can be estimated from the time autocorrelation of the uncontrolled solution.

- More investigation is required to assess the performance of the proposed preconditioned data assimilation method for more complex turbulent/atmospheric systems. The proposed method should first be applied to a 1D problem such as the Kuramoto Sivashinsky equation while assuming random errors in the measurements. The accuracy and computational cost should be compared with existing methods such as the standard Kalman filter.

- Regarding the preconditioning of the data assimilation Schur complement system, it was found that the smallest eigenvalue of the preconditioned system $\mu_{min} \rightarrow 0$ for small relaxation parameter values a. This may be avoided by modifying the preconditioner structure to improve the distribution of the smallest eigenvalues. For example, an improved preconditioner can be derived if a better approximation of the Schur complement matrix is formed. At the moment, only the diagonal blocks are retained, each diagonal block corresponding to a single segment. An idea would be to retain blocks that correspond to two consecutive segments, instead of just one. In such a case, the cost of applying the preconditioner would be higher, but if it results in a much faster convergence rate, then this overhead would be justified. Several tests need to be done to investigate the performance of such more advanced algorithms.

Bibliography

[1] Patrick J. Blonigan and Qiqi Wang. Multiple shooting shadowing for sensitivity analysis of chaotic dynamical systems. *Journal of Computational Physics*, 354:447–475, 2018.

[2] Antony Jameson. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3(3):233–260, 9 1988.

[3] Paolo Luchini and Alessandro Bottaro. Adjoint Equations in Stability Analysis. *Annual Review of Fluid Mechanics*, 46(1):493–517, 2014.

[4] Wei Liao and Her Mann Tsai. Aerodynamic shape optimization on overset grids using the adjoint method. *International Journal for Numerical Methods in Fluids*, 62(12):1332–1356, 2010.

[5] T Medjo and Louis Tebou. Adjoint-Based Iterative Method for Robust Control Problems in Fluid Mechanics. *SIAM J. Numerical Analysis*, 42:302–325, 2004.

[6] Dinesh Kumar, Mehrdad Raisee, and Chris Lacor. Combination of Polynomial Chaos with Adjoint Formulations for Optimization Under Uncertainties. In *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, pages 567–582. 2019.

[7] Thomas R. Bewley, Parviz Moin, and Roger Temam. DNS-based predictive control of turbulence: an optimal benchmark for feedback algorithms. *Journal of Fluid Mechanics*, 447:179–225, 2001.

[8] Bartosz Protas, Thomas R. Bewley, and Greg Hagen. A computational framework for the regularization of adjoint analysis in multiscale PDE systems. *Journal of Computational Physics*, 195(1):49–89, 2004.

[9] Xuerui Mao and Emily Pearson. Drag reduction and thrust generation by tangential surface motion in flow past a cylinder. *Theoretical and Computational Fluid Dynamics*, 32(3):307–323, 2018.

[10] Dandan Xiao and George Papadakis. Nonlinear optimal control of bypass transition in a boundary layer flow. *Physics of Fluids*, 29(5):054103, 2017.

[11] Dandan Xiao and George Papadakis. Nonlinear optimal control of transition due to a pair of vortical perturbations using a receding horizon approach. *Journal of Fluid Mechanics*, 861:524–555, 2019.

[12] Pierre Gauthier. Chaos and quadri-dimensional data assimilation: a study based on the Lorenz model. *Tellus A: Dynamic Meteorology and Oceanography*, 44(1):2–17, 1992.

[13] Wanglung Chung, John M. Lewis, S. Lakshmivarahan, and S. K. Dhall. A comparison of variational data assimilation and nudging using a simple dynamical system with chaotic behavior. In *Proceedings of the 1996 ACM symposium on Applied Computing - SAC '96*, volume Part F1287, pages 454–462, New York, New York, USA, 1996. ACM Press.

[14] D. J. Mcgillicuddy, D. R. Lynch, A. M. Moore, W. C. Gentleman, C. S. Davis, and C. J. Meise. An adjoint data assimilation approach to diagnosis of physical and biological controls on Pseudocalanus spp. in the Gulf of Maine–Georges Bank region. *Fisheries Oceanography*, 7(3/4):205–218, 1998.

[15] Minjie Xu, Kai Fu, and Xianqing Lv. Application of Adjoint Data Assimilation Method to Atmospheric Aerosol Transport Problems. *Advances in Mathematical Physics*, pages 1–14, 2017.

[16] Niles A. Pierce and Michael B. Giles. Adjoint recovery of superconvergent functionals from PDE approximations. *SIAM Review*, 42(2):247–264, 2000.

[17] David A. Venditti and David L. Darmofal. Grid adaptation for functional outputs: Application to two-dimensional inviscid flows. *Journal of Computational Physics*, 176(1):40–69, 2001.

[18] Daniel J. Lea, Thomas W. N. Haine, Myles R. Allen, and James A. Hansen. Sensitivity analysis of the climate of a chaotic ocean circulation model. *Quarterly Journal of the Royal Meteorological Society*, 128(586):2587–2605, 2002.

[19] Patrick J. Blonigan. Adjoint sensitivity analysis of chaotic dynamical systems with non-intrusive least squares shadowing. *Journal of Computational Physics*, 348:803–826, 2017.

[20] Angxiu Ni, Qiqi Wang, Pablo Fernández, and Chaitanya Talnikar. Sensitivity analysis on chaotic dynamical systems by Finite Difference Non-Intrusive Least Squares Shadowing (FD-NILSS). *Journal of Computational Physics*, 394:615–631, 2019.

[21] Patrick J Blonigan, Qiqi Wang, Eric J Nielsen, and Boris Diskin. Least Squares Shadowing Sensitivity Analysis of Chaotic Flow around a Two-Dimensional Airfoil. In *54th AIAA Aerospace Sciences Meeting*, pages 1–28, Reston, Virginia, 2016. American Institute of Aeronautics and Astronautics.

[22] Qiqi Wang, Rui Hu, and Patrick Blonigan. Least Squares Shadowing sensitivity analysis of chaotic limit cycle oscillations. *Journal of Computational Physics*, 267:210–224, 2014.

[23] Stefanie Günther, Nicolas R. Gauger, and Qiqi Wang. A framework for simultaneous aerodynamic design optimization in the presence of chaos. *Journal of Computational Physics*, 328:387–398, 2017.

[24] Yukiko S. Shimizu and Krzysztof Fidkowski. Output Error Estimation for Chaotic Flows. In *46th AIAA Fluid Dynamics Conference*, Reston, Virginia, 2016. American Institute of Aeronautics and Astronautics.

[25] Yukiko S. Shimizu and Krzysztof Fidkowski. Output-Based Error Estimation for Chaotic Flows Using Reduced-Order Modeling. In *2018 AIAA Aerospace Sciences Meeting*, Reston, Virginia, 2018. American Institute of Aeronautics and Astronautics.

[26] J. P. Eckmann and D. Ruelle. Ergodic theory of chaos and strange attractors. *Reviews of Modern Physics*, 57(3):617–656, 1985.

[27] Steven H. Strogatz. *Nonlinear Dynamics and Chaos.* Westview Press, 2nd edition, 2015.

[28] K. Geist, U. Parlitz, and W. Lauterborn. Comparison of Different Methods for Computing Lyapunov Exponents. *Progress of Theoretical Physics*, 83(5):875–893, 1990.

[29] Giancarlo Benettin, Luigi Galgani, Antonio Giorgilli, and Jean-Marie Strelcyn. Lyapunov Characteristic Exponents for smooth dynamical systems and for hamiltonian systems; A method for computing all of them. Part 2: Numerical application. *Meccanica*, 15(1):21–30, 3 1980.

[30] David Ruelle. A review of linear response theory for general differentiable dynamical systems. *Nonlinearity*, 22(4):855–870, 2009.

[31] J. Doyne Farmer and John J. Sidorowich. Optimal shadowing and noise reduction. *Physica D: Nonlinear Phenomena*, 47(3):373–392, 1991.

[32] Patrick J. Blonigan and Qiqi Wang. Probability density adjoint for sensitivity analysis of the Mean of Chaos. *Journal of Computational Physics*, 270:660–686, 2014.

[33] David Ruelle. Differentiation of SRB states. *Communications in Mathematical Physics*, 187(1):227–241, 1997.

[34] G. Gallavotti and E. G. D. Cohen. Dynamical ensembles in stationary states. *Journal of Statistical Physics*, 80(5-6):931–970, 1995.

[35] Angxiu Ni. Hyperbolicity, shadowing directions and sensitivity analysis of a turbulent three-dimensional flow. *Journal of Fluid Mechanics*, 863:644–669, 2019.

[36] Daniel J Lea, Myles R Allen, and Thomas W.N. Haine. Sensitivity analysis of the climate of a chaotic system. *Tellus A: Dynamic Meteorology and Oceanography*, 52(5):523–532, 2000.

[37] G. L. Eyink, T. W.N. Haine, and D. J. Lea. Ruelle's linear response formula, ensemble adjoint schemes and Lévy flights. *Nonlinearity*, 17(5):1867–1889, 2004.

[38] J. Thuburn. Climate sensitivities via a Fokker–Planck adjoint approach. *Quarterly Journal of the Royal Meteorological Society*, 131(605):73–92, 2005.

[39] Davide Lasagna. Sensitivity Analysis of Chaotic Systems Using Unstable Periodic Orbits. *SIAM Journal on Applied Dynamical Systems*, 17(1):547–580, 2018.

[40] Gary J Chandler and Rich R Kerswell. Invariant recurrent solutions embedded in a turbulent two-dimensional Kolmogorov flow. *Journal of Fluid Mechanics*, 722:554–595, 2013.

[41] D.V Anosov. On a class of invariant sets of smooth dynamical systems. In *Proceedings. 5th International Conference on Nonlinear Oscillations, Kiev (39-45)*, 1970.

[42] Davide Lasagna, Ati Sharma, and Johan Meyers. Periodic shadowing sensitivity analysis of chaotic systems. *Journal of Computational Physics*, 391:119–141, 2019.

[43] Angxiu Ni and Qiqi Wang. Sensitivity analysis on chaotic dynamical systems by Non-Intrusive Least Squares Shadowing (NILSS). *Journal of Computational Physics*, 347:56–77, 2017.

[44] Patrick Blonigan, Steven Gomez, and Qiqi Wang. Least Squares Shadowing for Sensitivity Analysis of Turbulent Fluid Flows. *52nd Aerospace Sciences Meeting*, pages 1–24, 2014.

[45] Patrick J. Blonigan and Qiqi Wang. Least squares shadowing sensitivity analysis of a modified Kuramoto-Sivashinsky equation. *Chaos, Solitons and Fractals*, 64(1):16–25, 2014.

[46] Patrick Blonigan and Qiqi Wang. Multigrid-in-time for sensitivity analysis of chaotic dynamical systems. *Numerical Linear Algebra with Applications*, 24(3):e1946, 2017.

[47] William H. Press, Saul A Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes*. Cambridge University Press, New York, 3rd edition, 2007.

[48] Karim Shawki and George Papadakis. A preconditioned Multiple Shooting Shadowing algorithm for the sensitivity analysis of chaotic systems. *Journal of Computational Physics*, 398, 2019.

[49] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2nd edition, 2003.

[50] Michele Benzi, Gene H. Golub, and Jörg Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005.

[51] Andy J. Wathen. Preconditioning. *Acta Numerica*, 24:329–376, 2015.

[52] Howard Elman, David Silvester, and Andy Wathen. *Finite Elements and Fast Iterative Solvers*. Oxford University Press, 2nd edition, 2014.

[53] Eleanor McDonald, Jennifer Pestana, and Andy Wathen. Preconditioning and Iterative Solution of All-at-Once Systems for Evolutionary Partial Differential Equations. *SIAM Journal on Scientific Computing*, 40(2):A1012–A1033, 2018.

[54] Yang Cao, Mei-Qun Jiang, and Ying-Long Zheng. A splitting preconditioner for saddle point problems. *Numerical Linear Algebra with Applications*, 18(5):875–895, 2011.

[55] Gene H. Golub, Chen Greif, and James M. Varah. An Algebraic Analysis of a Block Diagonal Preconditioner for Saddle Point Systems. *SIAM Journal on Matrix Analysis and Applications*, 27(3):779–792, 2005.

[56] Mansoor Rezghi and S. M. Hosseini. Lanczos based preconditioner for discrete ill-posed problems. *Computing*, 88(1-2):79–96, 2010.

[57] G. W. Stewart. A Krylov–Schur Algorithm for Large Eigenproblems. *SIAM Journal on Matrix Analysis and Applications*, 23(3):601–614, 2002.

[58] Iain Waugh, Simon Illingworth, and Matthew Juniper. Matrix-free continuation of limit cycles for bifurcation analysis of large thermoacoustic systems. *Journal of Computational Physics*, 240:225–247, 2013.

[59] Juan Sánchez and Marta Net. On the Multiple Shooting Continuation of Periodic Orbits by Newton-Krylov Methods. *International Journal of Bifurcation and Chaos*, 20(1):43–61, 2010.

[60] James Baglama and Lothar Reichel. Augmented Implicitly Restarted Lanczos Bidiagonalization Methods. *SIAM Journal on Scientific Computing*, 27(1):19–42, 2005.

[61] Colin Sparrow. *The Lorenz Equations: Bifurcations, Chaos, and Strange Attractors*, volume 41 of *Applied Mathematical Sciences*. Springer New York, New York, NY, 1982.

[62] Jan Frøyland and Knut H. Alfsen. Lyapunov-exponent spectra for the Lorenz model. *Physical Review A*, 29(5):2928–2931, 1984.

[63] Christopher L. Wolfe and Roger M. Samelson. An efficient method for recovering Lyapunov vectors from singular vectors. *Tellus, Series A: Dynamic Meteorology and Oceanography*, 59(3):355–366, 2007.

[64] Philip Holmes, John L. Lumley, Gahl Berkooz, and Clarence W. Rowley. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge University Press, Cambridge, 2nd edition, 2012.

[65] Shervan Erfani, Ali Tavakoli, and Davod Khojasteh Salkuyeh. An efficient method to set up a Lanczos based preconditioner for discrete ill-posed problems. *Applied Mathematical Modelling*, 37(20-21):8742–8756, 2013.

[66] Per Christian Hansen. Regularization Tools: A Matlab package for analysis and solution of discrete ill-posed problems. *Numerical Algorithms*, 6(1):1–35, 1994.

[67] D. Calvetti, G. H. Golub, and L. Reichel. Estimation of the L-Curve via Lanczos Bidiagonalization. *BIT Numerical Mathematics*, 39(4):603–619, 1999.

[68] D. Calvetti, L. Reichel, and A. Shuibi. L-curve and curvature bounds for Tikhonov regularization. *Numerical Algorithms*, 35(2-4):301–314, 2004.

[69] P. Fernandez and Q. Wang. Lyapunov spectrum of the separated flow around the NACA 0012 airfoil and its dependence on numerical discretization. *Journal of Computational Physics*, 350:453–469, 2017.

[70] Malik Hassanaly and Venkat Raman. Lyapunov spectrum of forced homogeneous isotropic turbulent flows. *arXiv*, 2019.

[71] Karim Shawki and George Papadakis. Feedback control of chaotic systems using multiple shooting shadowing and application to Kuramoto–Sivashinsky equation. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 476(2240), 2020.

[72] Onofrio Semeraro, Shervin Bagheri, Luca Brandt, and Dan S. Henningson. Feedback control of three-dimensional optimal disturbances using reduced-order models. *Journal of Fluid Mechanics*, 677:63–102, 2011.

[73] Alexandre Barbagallo, Denis Sipp, and Peter J. Schmid. Closed-loop control of an open cavity flow using reduced-order models. *Journal of Fluid Mechanics*, 641:1–50, 2009.

[74] Arthur Bryson, Y.-C Ho, and George Siouris. *Applied Optimal Control.* Hemisphere Pub. Corp, 1975.

[75] Thomas R. Bewley and Sharon Liu. Optimal and robust control and estimation of linear paths to transition. *Journal of Fluid Mechanics*, 365:305–349, 1998.

[76] Keun H. Lee, Luca Cortelezzi, John Kim, and Jason Speyer. Application of reduced-order controller to turbulent flows for drag reduction. *Physics of Fluids*, 13(5):1321–1330, 2001.

[77] J. McKernan, J. F. Whidborne, and G. Papadakis. Linear quadratic control of plane Poiseuille flow-the transient behaviour. *International Journal of Control*, 80(12):1912–1930, 2007.

[78] A. S. Sharma, J. F. Morrison, B. J. McKeon, D. J.N. Limebeer, W. H. Koberg, and S. J. Sherwin. Relaminarisation of $\mathrm{Re}\tau = 100$ channel flow with globally stabilising linear feedback control. *Physics of Fluids*, 23(12), 2011.

[79] Peter H. Heins, Bryn Ll Jones, and Ati S. Sharma. Passivity-based output-feedback control of turbulent channel flow. *Automatica*, 69:348–355, 2016.

[80] Shervin Bagheri, Luca Brandt, and Dan S. Henningson. Input–output analysis, model reduction and control of the flat-plate boundary layer. *Journal of Fluid Mechanics*, 620:263, 2009.

[81] John Kim and Thomas R. Bewley. A Linear Systems Approach to Flow Control. *Annual Review of Fluid Mechanics*, 39(1):383–417, 2007.

[82] Panagiotis D. Christofides and Antonios Armaou. Global stabilization of the Kuramoto-Sivashinsky equation via distributed output feedback control. *Systems and Control Letters*, 39(4):283–294, 2000.

[83] S. N. Gomes, M. Pradas, S. Kalliadasis, D. T. Papageorgiou, and G. a. Pavliotis. Controlling spatiotemporal chaos in active dissipative-dispersive nonlinear systems. *Physical Review E*, 92(2):022912, 2015.

[84] Susana N. Gomes, Demetrios T. Papageorgiou, and Grigorios A. Pavliotis. Stabilizing non-Trivial solutions of the generalized Kuramoto-Sivashinsky equation using feedback and optimal control. *IMA Journal of Applied Mathematics (Institute of Mathematics and Its Applications)*, 82(1):158–194, 2017.

[85] Onofrio Semeraro, Jan O. Pralits, Clarence W. Rowley, and Dan S. Henningson. Riccati-less approach for optimal control and estimation: An application to two-dimensional boundary layers. *Journal of Fluid Mechanics*, 731:394–417, 2013.

[86] Jan Oscar Pralits and Paolo Luchini. Riccati-less optimal control of bluff-body wakes. pages 325–330. 2010.

[87] Thomas Bewley, Paolo Luchini, and Jan Pralits. Methods for solution of large

optimal control problems that bypass open-loop model reduction. *Meccanica*, 51(12):2997–3014, 2016.

[88] Onofrio Semeraro and Jan O. Pralits. Full-order optimal compensators for flow control: the multiple inputs case. *Theoretical and Computational Fluid Dynamics*, 32(3):285–305, 2018.

[89] Karl Mårtensson. *Gradient Methods for Large-Scale and Distributed Linear Quadratic Control*. PhD thesis, 2012.

[90] Luiz V.R. Cagliari, Sandipan Mishra, and Jason E. Hicken. Plant and controller optimization in the context of chaotic dynamical systems. *AIAA Scitech 2019 Forum*, (January):1–21, 2019.

[91] Karl J. Astrom. *Introduction to Stochastic Control Theory*. Dover, New York, 2006.

[92] Richard P. Brent. *Algorithms for Minimization without Derivatives*. Prentice-Hall, 1973.

[93] R.R. Kerswell. Nonlinear Nonmodal Stability Theory. *Annual Review of Fluid Mechanics*, 50(1):319–345, 2018.

[94] Markus Hogberg, Thomas R. Bewley, and Dan S. Henningson. Linear feedback control and estimation of transition in plane channel flow. *Journal of Fluid Mechanics*, 481(481), 2003.

[95] Vishwas Rao and Adrian Sandu. A time-parallel approach to strong-constraint four-dimensional variational data assimilation. *Journal of Computational Physics*, 313:583–593, 2016.

[96] P. Courtier, J.-N. Thépaut, and A. Hollingsworth. A strategy for operational implementation of 4D-Var, using an incremental approach. *Quarterly Journal of the Royal Meteorological Society*, 120:1367–1387, 1994.

[97] F.-X. Le Dimet, H.-E. Ngodock, B. Luong, and J. Verron. Sensitivity Analysis in Variational Data Assimilation. *Journal of the Meteorological Society of Japan. Ser. II*, 75(1B):245–255, 1997.

[98] Michael Fisher and Selime Gürol. Parallelization in the time dimension of four-dimensional variational data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 143(703):1136–1147, 2017.

[99] Gene H. Golub and Chen Greif. On Solving Block-Structured Indefinite Linear Systems. *SIAM Journal on Scientific Computing*, 24(6):2076–2092, 2003.

[100] M. Fisher, S. Gratton, S. Gürol, Y. Trémolet, and X. Vasseur. Low rank updates in preconditioning the saddle point systems arising from data assimilation problems. *Optimization Methods and Software*, 33(1):45–69, 2018.

[101] Melina A. Freitag and Daniel L.H. Green. A low-rank approach to the solution of weak constraint variational data assimilation problems. *Journal of Computational Physics*, 357:263–281, 2018.

[102] Michele Alessandro Bucci, Onofrio Semeraro, Alexandre Allauzen, Guillaume Wisniewski, Laurent Cordier, and Lionel Mathelin. Control of chaotic systems by deep reinforcement learning. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 475(2231):20190351, 2019.

Appendices

A Discretisation of the Kuramoto Sivashinsky Equation (KSE)

Consider again the KSE,

$$\frac{\partial u}{\partial t} = -(u+c)\frac{\partial u}{\partial x} - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^4 u}{\partial x^4} \qquad x \in [0, L] \tag{A.1a}$$

$$u(0,t) = u(L,t) = 0 \tag{A.1b}$$

$$\left.\frac{\partial u}{\partial x}\right|_{x=0} = \left.\frac{\partial u}{\partial x}\right|_{x=L} = 0 \tag{A.1c}$$

The grid uses $N+2$ nodes $(i = 0, 1, 2, \ldots, N, N+1)$ with equal spacing $\delta x = {}^{L}/_{N+1}$. To enforce the Dirichlet boundary conditions (A.1b) at $x = 0$ and $x = L$, $\mathbf{u}_0 = 0$ and $\mathbf{u}_{N+1} = 0$. The Neumann boundary conditions (A.1c) at $x = 0$ and $x = L$ are enforced by introducing ghost nodes, such that $\mathbf{u}_{-1} = \mathbf{u}_1$ and $\mathbf{u}_{N+2} = \mathbf{u}_N$.

The spatial derivatives are discretised using 2^{nd} order approximations as follows:

$$\begin{aligned}
\left.\frac{\partial u}{\partial x}\right|_i &\approx \frac{\mathbf{u}_{i+1} - \mathbf{u}_{i-1}}{2\delta x} && (i = 1, 2, \ldots, N)\\
u\left.\frac{\partial u}{\partial x}\right|_i &\approx \frac{\mathbf{u}_{i+1}^2 - \mathbf{u}_{i-1}^2}{4\delta x} && (i = 1, 2, \ldots, N)\\
\left.\frac{\partial^2 u}{\partial x^2}\right|_i &\approx \frac{\mathbf{u}_{i+1} - 2\mathbf{u}_i + \mathbf{u}_{i-1}}{\delta x^2} && (i = 1, 2, \ldots, N)\\
\left.\frac{\partial^4 u}{\partial x^4}\right|_i &\approx \frac{\mathbf{u}_{i-2} - 4\mathbf{u}_{i-1} + 6\mathbf{u}_i - 4\mathbf{u}_{i+1} + \mathbf{u}_{i+2}}{\delta x^4} && (i = 2, 3, \ldots, N-1)
\end{aligned} \tag{A.2}$$

For the 4^{th} order derivative at $i = 1$ and $i = N$,

$$\begin{aligned}
\left.\frac{\partial^4 u}{\partial x^4}\right|_1 &\approx \frac{7\mathbf{u}_1 - 4\mathbf{u}_2 + \mathbf{u}_3}{\delta x^4}\\
\left.\frac{\partial^4 u}{\partial x^4}\right|_N &\approx \frac{7\mathbf{u}_N - 4\mathbf{u}_{N-1} + \mathbf{u}_{N-2}}{\delta x^4}
\end{aligned} \tag{A.3}$$

Using (A.2) and (A.3), the KSE (A.1) can be written in general ODE form

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}), \quad \mathbf{u}(0) = \mathbf{u}_0 \tag{A.4}$$

where $\mathbf{u}(t) = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_N \end{bmatrix}^T$ and $\mathbf{f}(\mathbf{u})$ is written as follows

$$
\begin{aligned}
\mathbf{f}_1 &= \left(\frac{2}{\delta x^2} - \frac{7}{\delta x^4}\right)\mathbf{u}_1 + \left(-\frac{2c+\mathbf{u}_2}{4\delta x} - \frac{1}{\delta x^2} + \frac{4}{\delta x^4}\right)\mathbf{u}_2 - \frac{1}{\delta x^4}\mathbf{u}_3 \\
\mathbf{f}_2 &= \left(\frac{2c+\mathbf{u}_1}{4\delta x} - \frac{1}{\delta x^2} + \frac{4}{\delta x^4}\right)\mathbf{u}_1 + \left(\frac{2}{\delta x^2} - \frac{6}{\delta x^4}\right)\mathbf{u}_2 + \left(-\frac{2c+\mathbf{u}_3}{4\delta x} - \frac{1}{\delta x^2} + \frac{4}{\delta x^4}\right)\mathbf{u}_3 \\
&\quad - \frac{1}{\delta x^4}\mathbf{u}_4 \\
\mathbf{f}_i &= -\frac{1}{\delta x^4}\mathbf{u}_{i-2} + \left(\frac{2c+\mathbf{u}_{i-1}}{4\delta x} - \frac{1}{\delta x^2} + \frac{4}{\delta x^4}\right)\mathbf{u}_{i-1} + \left(\frac{2}{\delta x^2} - \frac{6}{\delta x^4}\right)\mathbf{u}_i \\
&\quad + \left(-\frac{2c+\mathbf{u}_{i+1}}{4\delta x} - \frac{1}{\delta x^2} + \frac{4}{\delta x^4}\right)\mathbf{u}_{i+1} - \frac{1}{\delta x^4}\mathbf{u}_{i+2} \qquad (i = 3, 4, \dots, N-2) \\
\mathbf{f}_{N-1} &= -\frac{1}{\delta x^4}\mathbf{u}_{N-3} + \left(\frac{2c+\mathbf{u}_{N-2}}{4\delta x} - \frac{1}{\delta x^2} + \frac{4}{\delta x^4}\right)\mathbf{u}_{N-2} + \left(\frac{2}{\delta x^2} - \frac{6}{\delta x^4}\right)\mathbf{u}_{N-1} \\
&\quad + \left(-\frac{2c+\mathbf{u}_N}{4\delta x} - \frac{1}{\delta x^2} + \frac{4}{\delta x^4}\right)\mathbf{u}_N \\
\mathbf{f}_N &= -\frac{1}{\delta x^4}\mathbf{u}_{N-2} + \left(\frac{2c+\mathbf{u}_{N-1}}{4\delta x} - \frac{1}{\delta x^2} + \frac{4}{\delta x^4}\right)\mathbf{u}_{N-1} + \left(\frac{2}{\delta x^2} - \frac{7}{\delta x^4}\right)\mathbf{u}_N
\end{aligned}
\tag{A.5}
$$

B Computation of the matrix-vector product $\left(\gamma I + \mathbf{M_{BD}}_{(l)}^{(q)} S\right) \underline{\mathbf{z}}$ and the vector $A\underline{\mathbf{g}}$

To compute the MATVEC product $\left(\gamma I + \mathbf{M_{BD}}_{(l)}^{(q)} AA^T\right) \underline{\mathbf{z}}$ for an arbitrary vector $\underline{\mathbf{z}}$:

i Use Algorithm 2 (Chapter 2.4) to compute $AA^T\underline{\mathbf{z}} = \underline{\mathbf{x}}$.

ii For $i = 1 : P$, compute the products $\mathbf{M}_{(l),i}^{(q)}\mathbf{x}_i$, then form $\mathbf{M_{BD}}_{(l)}^{(q)} AA^T\underline{\mathbf{z}} = \left[\left(\mathbf{M}_{(l),1}^{(q)}\mathbf{x}_1\right)^T \quad \left(\mathbf{M}_{(l),2}^{(q)}\mathbf{x}_2\right)^T \quad \dots \quad \left(\mathbf{M}_{(l),P}^{(q)}\mathbf{x}_P\right)^T\right]^T$.

iii Compute
$\left(\gamma I + \mathbf{M_{BD}}_{(l)}^{(q)} AA^T\right) \underline{\mathbf{z}} = \gamma\underline{\mathbf{z}} + \left[\left(\mathbf{M}_{(l),1}^{(q)}\mathbf{x}_1\right)^T \quad \left(\mathbf{M}_{(l),2}^{(q)}\mathbf{x}_2\right)^T \quad \dots \quad \left(\mathbf{M}_{(l),P}^{(q)}\mathbf{x}_P\right)^T\right]^T$.

To compute the vector $A\underline{\mathbf{g}}$:

i Compute the vectors $\mathbf{g}_i$ (2.29) for $i = 1 : P$ by integrating

$$\frac{d\hat{\mathbf{w}}}{dt} = -\frac{\partial \mathbf{f}}{\partial \mathbf{u}}^T \hat{\mathbf{w}} - \frac{1}{T}\frac{\partial J}{\partial \mathbf{u}}$$

backward in time with the terminal conditions $\hat{\mathbf{w}}(t_i^-) = \frac{1}{T}\frac{(\overline{J^{(T)}} - J_i)\mathbf{h}_i}{\|\mathbf{h}_i\|_2^2}$ to obtain $\hat{\mathbf{w}}(t_{i-1}^+) = \mathbf{g}_i$.

ii Form the vector $\Phi_i\mathbf{g}_i$ by integrating

$$\frac{d\mathbf{v}'}{dt} - \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\mathbf{v}' = 0$$

forward in time with the initial conditions $\mathbf{v}'(t_{i-1}^+) = \mathbf{g}_i$ to obtain $\mathbf{v}'(t_i^-)$. Then $\Phi_i\mathbf{g}_i = P_{t_i}\mathbf{v}'(t_i^-)$.

iii Form $A\underline{\mathbf{g}} = -\left[(\Phi_1\mathbf{g}_1 - \mathbf{g}_2)^T \quad \dots \quad (\Phi_{P-1}\mathbf{g}_{P-1} - \mathbf{g}_P)^T \quad (\Phi_P\mathbf{g}_P)^T\right]^T$

C Derivation of the optimality system (5.4)

The minimisation statement (5.1) is repeated again below:

$$
\begin{aligned}
\underset{\mathbf{u}}{\text{Minimise}} \quad \overline{J(T)} &= \frac{1}{2T}\int_0^T (\mathbf{u}-\mathbf{u}_m)^T Q(\mathbf{u}-\mathbf{u}_m)\,dt \\
\text{subject to} \quad \mathbf{r}_\mathbf{u} &= \frac{d\mathbf{u}}{dt} - \mathbf{f}(\mathbf{u},t) = 0
\end{aligned}
\tag{C.1}
$$

The optimality conditions are derived by introducing the following Lagrangian function $\mathscr{L}(\mathbf{u},\boldsymbol{\lambda})$:

$$
\mathscr{L}(\mathbf{u},\boldsymbol{\lambda}) = \frac{1}{2T}\int_0^T (\mathbf{u}-\mathbf{u}_m)^T Q(\mathbf{u}-\mathbf{u}_m)\,dt - \frac{1}{T}\int_0^T \boldsymbol{\lambda}^T\left(\frac{d\mathbf{u}}{dt} - \mathbf{f}(\mathbf{u},t)\right)\,dt \tag{C.2}
$$

The first order conditions ${}^{\partial\mathscr{L}}/_{\partial\boldsymbol{\lambda}} = {}^{\partial\mathscr{L}}/_{\partial\mathbf{u}} = 0$ must be satisfied to obtain the optimality system. Setting ${}^{\partial\mathscr{L}}/_{\partial\boldsymbol{\lambda}} = 0$ simply recovers the constraint equation (5.4a). The condition ${}^{\partial\mathscr{L}}/_{\partial\mathbf{u}} = 0$ is equivalent to

$$
\lim_{\epsilon\to 0}\left(\frac{\mathscr{L}(\mathbf{u}+\epsilon\delta\mathbf{u},\boldsymbol{\lambda}) - \mathscr{L}(\mathbf{u},\boldsymbol{\lambda})}{\epsilon}\right) = 0 \tag{C.3}
$$

Applying (C.3) to the right hand side of (C.2),

$$
\frac{\partial\mathscr{L}}{\partial\mathbf{u}} = \frac{1}{T}\int_0^T (\mathbf{u}-\mathbf{u}_m)^T Q\delta\mathbf{u}\,dt - \frac{1}{T}\int_0^T \boldsymbol{\lambda}^T\Big(\dot{\delta\mathbf{u}} - \big(\mathbf{f}(\mathbf{u}+\epsilon\delta\mathbf{u}) - \mathbf{f}(\mathbf{u})\big)\Big)\,dt = 0
$$

Substituting $\mathbf{f}(\mathbf{u}+\epsilon\delta\mathbf{u}) - \mathbf{f}(\mathbf{u}) \approx \frac{\partial\mathbf{f}}{\partial\mathbf{u}}\big|_\mathbf{u}\epsilon\delta\mathbf{u}$,

$$
\frac{\partial\mathscr{L}}{\partial\mathbf{u}} = \frac{1}{T}\int_0^T (\mathbf{u}-\mathbf{u}_m)^T Q\delta\mathbf{u}\,dt - \frac{1}{T}\int_0^T \boldsymbol{\lambda}^T\left(\dot{\delta\mathbf{u}} - \left.\frac{\partial\mathbf{f}}{\partial\mathbf{u}}\right|_\mathbf{u}\delta\mathbf{u}\right)\,dt = 0 \tag{C.4}
$$

Applying integration by parts to $\frac{1}{T}\int_0^T \boldsymbol{\lambda}^T\dot{\delta\mathbf{u}}\,dt$,

$$
\frac{1}{T}\int_0^T \boldsymbol{\lambda}^T\dot{\delta\mathbf{u}}\,dt = \boldsymbol{\lambda}^T(T)\delta\mathbf{u}(T) - \boldsymbol{\lambda}^T(0)\delta\mathbf{u}(0) - \frac{1}{T}\int_0^T \dot{\boldsymbol{\lambda}}^T\delta\mathbf{u}\,dt \tag{C.5}
$$

Since the variations $\delta\mathbf{u}(T)$ and $\delta\mathbf{u}(0)$ are arbitrary, the boundary conditions

$$\boldsymbol{\lambda}(0) = \boldsymbol{\lambda}(T) = 0$$

must be satisfied (equation 5.4c). Substituting (C.5) back into (C.4) gives

$$\frac{\partial \mathscr{L}}{\partial \mathbf{u}} = \frac{1}{T}\int_0^T (\mathbf{u} - \mathbf{u}_m)^T Q \delta\mathbf{u}\, dt + \frac{1}{T}\int_0^T \dot{\boldsymbol{\lambda}}^T \delta\mathbf{u} + \boldsymbol{\lambda}^T \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\mathbf{u}} \delta\mathbf{u}\, dt = 0$$

The collected terms with $\delta\mathbf{u}$ must be equal to zero, i.e.,

$$\frac{d\boldsymbol{\lambda}}{dt} + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\mathbf{u}}^T \boldsymbol{\lambda} + Q(\mathbf{u} - \mathbf{u}_m) = 0 \tag{C.6}$$

which is the adjoint equation (5.4b).

If the objective function of (C.1) is replaced by the more general form

$$\langle J^{(T)} \rangle = \frac{1}{2T}\int_0^T (C\mathbf{u}(t) - \mathbf{y}_m)^T (C\mathbf{u}(t) - \mathbf{y}_m)\, dt$$

then by following the same steps as above, the following optimality system is derived:

$$\frac{d\mathbf{u}}{dt} - \mathbf{f}(\mathbf{u}, t) = 0 \tag{C.7a}$$

$$\frac{d\boldsymbol{\lambda}}{dt} + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\mathbf{u}}^T \boldsymbol{\lambda} + C^T(C\mathbf{u} - \mathbf{y}_m) = 0 \tag{C.7b}$$

$$\boldsymbol{\lambda}(0) = \boldsymbol{\lambda}(T) = 0 \tag{C.7c}$$

The matrix C (output matrix) is size $M \times N$ and $\mathbf{y}_m$ is a vector of length M (number of available measurements). The terms $Q(\mathbf{u} - \mathbf{u}_m)$ in (C.6) and $C^T(C\mathbf{u} - \mathbf{y}_m)$ in (C.7b) are equivalent when recovering unmeasured states. The two optimality systems (5.4) and (C.7) are therefore identical for this case.

D Computation of the matrix-vector product $\left(\gamma I + \mathbf{M_{BD}}_{(l)}^{(q)} S\right) \underline{\mathbf{z}}$ and the vector $\left(A\tilde{G}^{-1}\underline{\mathbf{b}}_{\delta\boldsymbol{\lambda}} + \underline{\mathbf{b}}_{\delta\mathbf{u}}\right)$

To compute the MATVEC product $\left(\gamma I + \mathbf{M_{BD}}_{(l)}^{(q)} S\right) \underline{\mathbf{z}} = \left(\gamma I + \mathbf{M_{BD}}_{(l)}^{(q)} A\tilde{G}^{-1}A^T\right) \underline{\mathbf{z}}$ for an arbitrary vector $\underline{\mathbf{z}}$:

1. Compute $\underline{\mathbf{x}} = A^T\underline{\mathbf{z}}$: Integrate

$$\frac{d\mathbf{w}}{dt} + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}^T \mathbf{w} = 0$$

   backward in time in all segments $(i = 1, 2, ..., P)$ with the terminal conditions $\mathbf{w}(t_i^-) = \mathbf{z}_i$. This gives $\mathbf{w}(t_{i-1}^+) = \phi_i^T \mathbf{z}_i$. Form $\mathbf{x}_{i-1} = \mathbf{z}_{i-1} - \mathbf{w}(t_{i-1}^+)$ (for $i = 2, 3, ..., P-1$).

   Note that $\mathbf{x}_0 = -\mathbf{w}(t_0^+)$ and $\mathbf{x}_P = \mathbf{z}_P$.

2. Compute $\mathbf{v}_i = \check{Q}^{-1}\mathbf{x}_i$ (for $i = 0, 1, \ldots, P-1$) and $\mathbf{v}_P = (\epsilon^{-1}I)\mathbf{x}_P$.

3. Compute $\underline{\mathbf{y}} = A\tilde{G}^{-1}A^T\underline{\mathbf{z}} = A\underline{\mathbf{v}}$: Integrate

$$\frac{d\mathbf{v}'}{dt} - \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\mathbf{v}' = 0$$

   forwards in time in all segments with the initial conditions $\mathbf{v}'(t_{i-1}^+) = \mathbf{v}_{i-1}$ to obtain $\mathbf{v}'(t_i^-)$. Form $\underline{\mathbf{y}} = \left[\left(-\mathbf{v}'(t_1^-) + \mathbf{v}_1\right)^T \;\; \ldots \;\; \left(-\mathbf{v}'(t_P^-) + \mathbf{v}_P\right)^T\right]^T$.

4. For $i = 1 : P$, compute the products $\mathbf{M}_{(l),i}^{(q)}\mathbf{y}_i$, then form $\mathbf{M_{BD}}_{(l)}^{(q)} A\tilde{G}^{-1}A^T\underline{\mathbf{z}} = \left[\left(\mathbf{M}_{(l),1}^{(q)}\mathbf{y}_1\right)^T \;\; \left(\mathbf{M}_{(l),2}^{(q)}\mathbf{y}_2\right)^T \;\; \ldots \;\; \left(\mathbf{M}_{(l),P}^{(q)}\mathbf{y}_P\right)^T\right]^T$

5. Compute
   $\left(\gamma I + \mathbf{M_{BD}}_{(l)}^{(q)} S\right) \underline{\mathbf{z}} = \gamma\underline{\mathbf{z}} + \left[\left(\mathbf{M}_{(l),1}^{(q)}\mathbf{y}_1\right)^T \;\; \left(\mathbf{M}_{(l),2}^{(q)}\mathbf{y}_2\right)^T \;\; \ldots \;\; \left(\mathbf{M}_{(l),P}^{(q)}\mathbf{y}_P\right)^T\right]^T$

To compute the vector $A\tilde{G}^{-1}\underline{\mathbf{b}}_{\delta\boldsymbol{\lambda}} + \underline{\mathbf{b}}_{\delta\mathbf{u}}$:

1. Compute $\underline{\mathbf{b}}_{\delta\mathbf{u}}$: Integrate
$$\frac{d\mathbf{v}}{dt} - \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\mathbf{v} + \mathbf{r_u} = 0$$
forward in time in all segments with the initial conditions $\mathbf{v}(t_{i-1}^+) = 0$ to obtain $\mathbf{v}(t_i^-)$. Form $\underline{\mathbf{b}}_{\delta\mathbf{u}} = \begin{bmatrix} \mathbf{v}^T(t_1^-) & \mathbf{v}^T(t_1^-) & \ldots & \mathbf{v}^T(t_P^-) \end{bmatrix}^T$.

2. Compute $\underline{\mathbf{b}}_{\delta\boldsymbol{\lambda}}$: Integrate
$$\frac{d\mathbf{w}}{dt} + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}^T \mathbf{w} + \mathbf{r}_{\boldsymbol{\lambda}} = 0$$
backward in time in all segments with the terminal conditions $\mathbf{w}(t_i^-) = 0$ to obtain $\mathbf{w}(t_{i-1}^+)$.
Form $\underline{\mathbf{b}}_{\delta\boldsymbol{\lambda}} = \begin{bmatrix} \left(\boldsymbol{\lambda}_0^{(i)} - \mathbf{w}(t_0^+)\right)^T & -\left(\mathbf{w}(t_1^+)\right)^T & \ldots & -\left(\mathbf{w}(t_{P-1}^+)\right)^T & -\left(\boldsymbol{\lambda}_P^{(i)}\right)^T \end{bmatrix}^T$.

3. Compute $\underline{\mathbf{y}} = \tilde{G}^{-1}\underline{\mathbf{b}}_{\delta\boldsymbol{\lambda}} =$
$\begin{bmatrix} \left(\check{Q}^{-1}\left(\boldsymbol{\lambda}_0^{(i)} - \mathbf{w}(t_0^+)\right)\right)^T & -\left(\check{Q}^{-1}\mathbf{w}(t_1^+)\right)^T & \ldots & -\left(\check{Q}^{-1}\mathbf{w}(t_{P-1}^+)\right)^T & -\left(\epsilon^{-1} I \boldsymbol{\lambda}_P^{(i)}\right)^T \end{bmatrix}^T$.

4. Compute $A\underline{\mathbf{y}} = A\tilde{G}^{-1}\underline{\mathbf{b}}_{\delta\boldsymbol{\lambda}}$: Integrate
$$\frac{d\mathbf{v}'}{dt} - \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\mathbf{v}' = 0$$
forward in time in all segments with the initial conditions $\mathbf{v}'(t_{i-1}^+) = \mathbf{y}_{i-1}$ to obtain $\mathbf{v}'(t_i^-)$. Form $A\underline{\mathbf{y}} = \begin{bmatrix} \left(-\mathbf{v}'(t_1^-) + \mathbf{y}_1\right)^T & \ldots & \left(-\mathbf{v}'(t_P^-) + \mathbf{y}_P\right)^T \end{bmatrix}^T$.

5. Form $A\underline{\mathbf{y}} + \underline{\mathbf{b}}_{\delta\mathbf{u}} = A\tilde{G}^{-1}\underline{\mathbf{b}}_{\delta\boldsymbol{\lambda}} + \underline{\mathbf{b}}_{\delta\mathbf{u}}$.