**National Technical University of Athens**
**School of Mechanical Engineering**
**Fluids Section**
**Parallel CFD & Optimization Unit**

# Software Programming for the Analysis of Chaotic Systems using Least Squares Shadowing, Data Assimilation and Gridless Flow Analysis using Neural Networks

Diploma Thesis

**Georgios D. Vamvouras**

Advisor: Kyriakos C. Giannakoglou, Professor NTUA

Athens, 2025

# Acknowledgments

I would like to express my sincere gratitude to my supervisor, Professor Kyriakos C. Giannakoglou, for his unwavering support throughout my Diploma Thesis journey and for granting me the opportunity to work under his supervision for the better part of two years. Thanks to his mentorship, I managed to attain new skills and sharpen my existing ones, inspired by his relentless pursuit of perfection and awe-inspiring dedication to his work, of which I hope to have picked up even a little, along the way. His analytical proficiency fundamentally enhanced my approach to problem-solving, not only within the context of scientific research, but life in general.

Additionally, I would like to thank the members of PCOpt/NTUA that were more than willing to assist me with anything I needed. However, special thanks go to Dr. Varvara Asouti for her valuable advice, which aided me in completing this thesis, as well as other projects I undertook before it, and for the time she so generously invested in doing so.

I also feel the need to thank God for giving me the strength to overcome the many obstacles I faced during my studies and for blessing me with my amazing parents, Marina and Dimitris, whose sacrifices I can only hope to one day be worthy of.

Last but certainly not least, I am deeply grateful to the person who has stood by me with unwavering support—my loving partner, Marina—who was there for me every step of the way and assisted me in the most selfless manner in everything I needed. Her presence has been both my strength and my solace, and for that, I am endlessly thankful.

National Technical University of Athens
School of Mechanical Engineering
Fluids Section
Parallel CFD & Optimization Unit

# Software programming for the analysis of chaotic systems using least squares shadowing, data assimilation and gridless flow analysis using neural networks

Diploma Thesis

**Georgios D. Vamvouras**

Advisor: Kyriakos C. Giannakoglou, Professor NTUA

Athens, 2025

## Abstract

This diploma thesis explores three emerging computational methodologies that are rapidly gaining attention in the fields of sensitivity analysis, data assimilation and artificial intelligence. It focuses on the Least Squares Shadowing (LSS) algorithm for sensitivity analysis, data assimilation for improved accuracy of state estimation, and Physics-Informed Neural Networks (PINNs) for gridless flow simulations.

Traditional gradient-based sensitivity analysis techniques, such as finite differences and adjoint methods, often fail when applied to chaotic systems due to their extreme sensitivity to initial conditions, which leads to the exponential divergence of trajectories with infinitesimally small difference of parameters. The LSS method is shown to be a stable and computationally efficient alternative, leveraging the mathematical properties of dynamical systems encountered in modelling of physical phenomena, to ensure that perturbations in design parameters produce meaningful sensitivity derivatives. By reformulating sensitivity analysis as a constrained optimization problem, LSS allows for accurate derivative computations, even in highly chaotic regimes. Also, a Discretely Consistent LSS (DCLSS) formulation is developed, improving numerical accuracy by ensuring consistency in finite difference discretization schemes. The effectiveness of these methods is demonstrated through applications to three mathematical problems, where conventional approaches fail to provide reliable results, preparing the ground for CFD applications.

Beyond sensitivity analysis, this thesis examines the role of data assimilation in enhancing model prediction accuracy during unsteady simulations. A data assimilation technique, namely the Extended Kalman Filter (EKF), is used to incorporate noisy experimental data or observations into simulations based on noisy models of dynamical systems with mathematically proven optimality, thereby producing states more accurate than either the model or the observations alone. This process practically

corrects model errors by combining numerical predictions and observational data. Applications to the mathematical systems illustrate the EKF's ability to improve prediction accuracy, even in the presence of incomplete or uncertain measurements. Additionally, parametric studies assess the impact of key hyper-parameters on the effectiveness of data assimilation, providing insights into its robustness and practical implementation.

Finally, the diploma thesis explores the use of PINNs as numerical solvers for fluid flow simulations, eliminating the need for discrete computational grids. PINNs embed governing equations, boundary conditions, and physical constraints directly into the loss function of a neural network, leveraging automatic differentiation to compute derivatives efficiently. Two flow problems are considered: a steady, incompressible quasi-1D duct flow and a laminar flow through a 2D duct. The results demonstrate that PINNs can successfully capture complex flow behavior while providing smooth, continuous solutions. Compared to conventional CFD methods, PINNs offer a promising approach for solving PDEs in irregular geometries, though their computational efficiency and accuracy leave room for improvement.

Overall, this diploma thesis accomplices to utilize the LSS method of computing SDs, in cases where adjoint methods fail and FDs are prohibitively expensive, such as many chaotic and/or challenging systems, successfully incorporating them in optimization loops. Moreover, the EKF is utilized as an efficient way to incorporate experimental data in numerical simulations, correcting the unavoidable uncertainties contained in models of complex phenomena, which is especially useful in chaotic systems. Finally, PINNs are shown capable of solving complex PDE systems, by incorporating their physics in the network loss function. All three of these methodologies are demonstrated by numerous cases developed in Python and C++ codes, and a thorough examination of their nature is performed.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Μηχανολόγων Μηχανικών
Τομέας Ρευστών
Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής & και Βελτιστοποίησης

# Προγραμματισμός Λογισμικού για την Ανάλυση Χαοτικών Συστημάτων με Μεθόδους Σκίασης Ελαχίστων Τετραγώνων, την Αφομοίωση Δεδομένων και την Επίλυση των Εξισώσεων Ροής Χωρίς Πλέγματα με Νευρωνικά Δίκτυα

Διπλωματική Εργασία

**Γεώργιος Δ. Βάμβουρας**

Επιβλέπων: Κυριάκος Χ. Γιαννάκογλου, Καθηγητής ΕΜΠ

Αθήνα, 2025

## Περίληψη

Στα πλαίσια αυτής της διπλωματικής εργασίας διερευνώνται τρεις αναδυόμενες υπολογιστικές μεθοδολογίες που προσελκύουν ραγδαία το ενδιαφέρον στους τομείς της Ανάλυσης Ευαισθησίας (Sensitivity Analysis), της Αφομοίωσης Δεδομένων (Data Assimilation, DA) και της Τεχνητής Νοημοσύνης (Artificial Intelligence, AI). Αρχικά παρουσιάζεται η μέθοδος Σκίασης Ελαχίστων Τετραγώνων (Least Squares Shadowing, LSS) για την ανάλυση ευαισθησίας, η αφομοίωση δεδομένων (Data Assimilation, DA) για τη βελτίωση της ακρίβειας της εκτίμησης καταστάσεων συστημάτων, και τα Ενημερωμένα από τη Φυσική των Ροών Νευρωνικά Δίκτυα (Physics-Informed Neural Networks, PINNs) για προσομοιώσεις ροής χωρίς χρήση πλέγματος.

Συμβατικές μέθοδοι ανάλυσης ευαισθησίας, όπως οι πεπερασμένες διαφορές και οι συζυγείς μέθοδοι (adjoint methods), συχνά αποτυγχάνουν όταν εφαρμόζονται σε χαοτικά συστήματα, λόγω της ακραίας ευαισθησίας τους στις αρχικές συνθήκες και στις τιμές των παραμέτρων, η οποία οδηγεί σε εκθετική απόκλιση των τροχιών ακόμα και για απειροελάχιστες μεταβολές των παραμέτρων. Η μέθοδος LSS αποτελεί μια σταθερή και υπολογιστικά αποδοτική εναλλακτική, καθώς διασφαλίζει ότι οι απειροστές μεταβολές στις μεταβλητές σχεδιασμού οδηγούν σε απειροστές μεταβολές στις τροχιές, και επομένως μπορούν να χρησιμοποιηθούν για την εύρεση παραγώγων ευαισθησίας, μέσω της αναδιατύπωσης της ανάλυσης ευαισθησίας ως ένα πρόβλημα βελτιστοποίησης υπό περιορισμούς. Έτσι, η LSS επιτρέπει ακριβείς υπολογισμούς παραγώγων ακόμη και σε έντονα χαοτικά συστήματα. Επιπλέον, αναπτύσσεται μια Διακριτά Συνεπής διατύπωση της LSS, (Discretely Consistent LSS, DCLSS), η οποία βελτιώνει την ακρίβεια και εξασφαλίζοντας συνέπεια στα σχήματα διακριτοποίησης πεπερασμένων διαφορών, ενώ επιτυγχάνει να υπολογίσει παραγώγους ευαισθησίας ακόμα και σε περιπτώσεις που

η LSS αποτυγχάνει. Η αποτελεσματικότητα αυτών των μεθόδων αποδεικνύεται μέσα από εφαρμογές σε τρία μαθηματικά προβλήματα, όπου οι συμβατικές προσεγγίσεις αποτυγχάνουν να παρέχουν αξιόπιστα αποτελέσματα, προετοιμάζοντας το έδαφος για εφαρμογές στην Υπολογιστική Ρευστοδυναμική (CFD).

Πέρα από την ανάλυση ευαισθησίας, η εργασία εξετάζει τον ρόλο της αφομοίωσης δεδομένων στη βελτίωση της ακρίβειας των προβλέψεων μοντέλων κατά τη διάρκεια μη μόνιμων προσομοιώσεων. Μια τεχνική αφομοίωσης δεδομένων, συγκεκριμένα το Εκτεταμένο Φίλτρο Kalman (Extended Kalman Filter, EKF), χρησιμοποιείται για την ενσωμάτωση θορυβωδών πειραματικών δεδομένων (παρατηρήσεων) σε προσομοιώσεις βασισμένες σε θορυβώδη μοντέλα, διασφαλίζοντας μαθηματικά αποδεδειγμένη βέλτιστη ακρίβεια. Με αυτόν τον τρόπο, τα σφάλματα του μοντέλου διορθώνονται στην πορεία της προσομοίωσης, συνδυάζοντας 'ζωντανά ' αριθμητικές προβλέψεις και δεδομένα παρατήρησης. Οι εφαρμογές σε μαθηματικά συστήματα δείχνουν ότι το EKF μπορεί να βελτιώσει σημαντικά την ακρίβεια των προβλέψεων, ακόμη και παρουσία πολύ λίγων μετρήσεων. Επιπλέον, πραγματοποιούνται παραμετρικές μελέτες για την αξιολόγηση της επίδρασης βασικών υπέρ-παραμέτρων στην αποτελεσματικότητα της μεθόδου.

Τέλος, η διπλωματική εργασία εξετάζει τη χρήση των PINNs ως αριθμητικών επιλυτών για προσομοιώσεις ροής ρευστών, εξαλείφοντας την ανάγκη για διακριτά υπολογιστικά πλέγματα. Τα PINNs ενσωματώνουν τις εξισώσεις, τις συνοριακές συνθήκες και φυσικούς περιορισμούς απευθείας στη συνάρτηση κόστους ενός νευρωνικού δικτύου, χρησιμοποιώντας αυτόματη διαφόριση (automatic differentiation) για υπολογισμούς παραγώγων χωρίς σχήματα διακριτοποίησης. Εξετάζονται δύο προβλήματα ροής: μια μόνιμη, ασυμπίεστη ψευδο-1D ροή σε αγωγό μεταβλητής διατομής, και μια στρωτή ροή μέσα σε 2D αγωγό μεταβλητής διατομής. Τα αποτελέσματα δείχνουν ότι τα PINNs μπορούν επιτυχώς να επιλύσουν πολύπλοκες ροές, παράγοντας αναλυτικές λύσεις. Σε σύγκριση με τις συμβατικές μεθόδους CFD, τα PINNs προσφέρουν μια πολλά υποσχόμενη προσέγγιση για την επίλυση μερικών διαφορικών εξισώσεων (PDEs) σε περίπλοκες γεωμετρίες, αν και η υπολογιστική τους αποδοτικότητα και ακρίβεια αφήνουν περιθώριο για βελτίωση.

Συνολικά, η παρούσα διπλωματική εργασία αναλύει τη μέθοδο LSS για τον υπολογισμό των ΣΔς, σε περιπτώσεις όπου οι συζυγείς μέθοδοι αποτυγχάνουν και οι πεπερασμένες διαφορές γίνονται απαγορευτικά δαπανηρές, όπως συμβαίνει σε πολλά χαοτικά ή/και απαιτητικά συστήματα, ενσωματώνοντάς τις επιτυχώς σε βρόχους βελτιστοποίησης. Επιπλέον, το EKF αξιοποιείται ως αποτελεσματική μέθοδος ενσωμάτωσης πειραματικών δεδομένων σε αριθμητικές προσομοιώσεις, διορθώνοντας τις αναπόφευκτες αβεβαιότητες που υπάρχουν στα μοντέλα πολύπλοκων φαινομένων, κάτι που είναι ιδιαίτερα έκδηλο σε χαοτικά συστήματα. Τέλος, αποδεικνύεται ότι τα PINNs είναι ικανά να επιλύουν πολύπλοκα συστήματα ΜΔΕ, ενσωματώνοντας τη φυσική τους στη συνάρτηση απωλειών του δικτύου. Και οι τρεις αυτές μεθοδολογίες επιδεικνύονται μέσω πολυάριθμων εφαρμογών που αναπτύχθηκαν σε κώδικες Python και C++, ενώ πραγματοποιείται μια διεξοδική ανάλυση των δυνατοτήτων τους.

# Contents

# Chapter 1

# Introduction

## 1.1 Gradient-Based Sensitivity Analysis

Gradient-based sensitivity analysis is a powerful tool that plays a crucial role in the design and optimization of complex dynamical systems. This analytical technique focuses on computing the sensitivities of specific quantities of interest—often referred to as objective functions—relative to various design parameters. By understanding how changes in design parameters influence the performance of a system, informed decisions can be made, that lead to optimized designs.

Two common approaches employed in sensitivity analysis are the Finite Difference (FD) method and (Continuous or Discrete) Adjoint method. The Finite Difference method estimates the Sensitivity Derivative (SD) by calculating the difference in the quantity of interest for an original design parameter value and a slightly perturbed value. This difference is then divided by the magnitude of the perturbation, yielding an approximation to the sensitivity. While this method is straightforward and easy to implement, it may require multiple evaluations of the objective function, which can be computationally expensive, especially for complex systems.

On the other hand, the Adjoint method is particularly advantageous when dealing with problems that involve multiple design parameters. This approach formulates the sensitivity analysis in terms of adjoint equations, allowing for the simultaneous computation of sensitivities with respect to all design parameters in a single run. As a result, the Adjoint method is often proven to be more efficient and less demanding in terms of computational resources compared to the Finite Difference method and other popular sensitivity analysis techniques. By leveraging the mathematical properties of adjoint equations, this method not only reduces the computational cost but also enhances the accuracy of sensitivity predictions, making it the preferred choice

in almost all cases.

## 1.2 Failure of Conventional Sensitivity Analysis Methods in Chaotic Problems

In cases in which the objective function represents a long-time average of chaotic problem solutions, the Finite Difference (FD) method becomes impractically costly, and the Continuous Adjoint (CA) method often fails altogether, resulting in meaningless SD values. This limitation poses significant challenges in real-life applications that involve physical phenomena exhibiting chaotic dynamics. For instance, in simulations of chaotic flows, such as those encountered in aerodynamics, the objective function might be defined as the long-time average of lift or drag forces acting on an object, such as an airfoil. The inherent chaotic nature of these systems means that even infinitesimally small changes in initial conditions or design parameters can lead to dramatically different outcomes. This sensitivity to initial conditions characteristic of chaos, leads to a tendency of small perturbations to escalate and amplify, causing the solutions to diverge exponentially over time.

## 1.3 The Least Squares Shadowing (LSS) Algorithm

In order for sensitivity analysis method to yield valid results, small changes in the initial conditions and design variables need to amount to small changes in the solution and consequently the objective function. This necessity is satisfied in the Least Squares Shadowing (LSS) algorithm, proposed in [5, 33, 34, 26, 4, 10, 6, 7, 3, 19, 32], which overcomes these problems and produces precise SD values with limited computational cost. It is based on the ergodicity of such systems, a property of most natural systems, which implies that for a sufficiently large averaging interval, the objective function is independent of the initial conditions. This allows for the use of different initial conditions for the solution of the problem with the perturbed design parameter, evaluated by means of a least-squares minimization problem that guarantees the proximity of the non-perturbed and perturbed parameter solutions (trajectories) for the entire time interval. A meaningful comparison of the trajectories can be made then, a result of which is the correct SD value.

In this thesis, the failure of conventional sensitivity methods to produce accurate SDs and the success of the LSS method is demonstrated using two small chaotic systems (Lorenz 1963 and Rössler equations), and another non-chaotic though challenging system, the Van Der Pol equations. It is shown that the Adjoint method collapses entirely and the FD method, although it does not collapse, becomes extremely inefficient and costly, which in larger problems would render it unusable. The LSS

algorithm is then used to produce valid SDs at limited cost, and the effect of some relevant hyper parameters is studied. Also, a Discretely Consistent version of the LSS algorithm (DCLSS) is developed which is proven to be more accurate in most test cases. The DCLSS algorithm guarantees that the FD schemes used to discretize temporal derivatives in the governing equations, are consistent with each other, which results to increased accuracy and precision.

## 1.4 Data Assimilation

The potential of data assimilation methods to overcome the issues caused by the unavoidable uncertainties in models of physical phenomena, is also examined in this thesis. Complex systems, such as turbulent flow, are especially hard to model as they are inherently non-linear and extremely sensitive to initial conditions, making accurate predictions challenging to achieve. Data assimilation aims to improve the prediction accuracy of such models by integrating measurements or experimental data into the procedure, by means that utilize the information provided by the model and observations in a most efficient way. The uncertainty of the model and the observational data is managed so that the assimilated state that is produced is more accurate than either of those. The data assimilation process examined in this diploma thesis is called filtering, and essentially introduces data in the simulation while the solution is propagated forward in time, in order to correct the model prediction at run time. The Extended Kalman Filter is applied to the Lorenz 1963, Rössler and Van der Pol systems, and its ability to overcome the model and observation errors (introduced manually for the purpose of demonstration) is successfully showcased. It is worth mentioning that DA is also applicable to chaotic systems, as shown in the Lorenz 1963 and Rössler cases, and it manages to overcome their massive unpredictability. Furthermore, parametric studies are preformed in order to examine the dependency of the assimilated state's accuracy to several hyper parameters.

## 1.5 Flow Solution Using Physics-Informed Neural Networks

Physics-Informed Neural Networks (PINNs) have emerged as a promising tool for solving Ordinary Differential Equations (ODEs) and Partial Differential Equations (PDEs), by embedding physical laws directly into the neural network loss function [28]. First introduced in [28] (2017-2019), PINNs leverage automatic differentiation to compute derivatives efficiently and enforce governing equations, boundary conditions, and initial conditions during training. Unlike traditional numerical

methods, which rely on discretized grids, PINNs provide a grid-less alternative that can generalize across different problem settings [22]. The application of PINNs to fluid dynamics and flow-related problems has gained significant attention due to their ability to handle irregular geometries and provide smooth and continuous solutions [8]. However, challenges such as convergence difficulties, boundary condition enforcement, and computational cost remain key limitations [29]. Recent advancements, including adaptive learning strategies [24], domain decomposition [15], and hybrid approaches combining PINNs with numerical solvers [25], have improved their accuracy and robustness [1, 23, 35, 2, 14]. This thesis explores the use of PINNs for solving the PDEs governing two types of flow. Firstly, a pseudo-1D, steady, inviscid flow of an incompressible fluid through an axisymetric duct of varying cross-sectional area is solved, along with the adjoint equations derived using the continuous adjoint method within the context of a shape optimization loop. Then, a steady, 2D, laminar flow of an incompressible fluid through a duct of varying cross-sectional area is solved, and compared to the results from in-house GPU-enabled CFD code PUMA
.

# Chapter 2

# Least Squares Shadowing (LSS): Application to the Lorenz '63 Problem

In this chapter, the Least Squares Shadowing (LSS) method for evaluation of the SDs of long-time averaged quantities w.r.t. a design variable in chaotic problems, is presented. For clarity, and in order to avoid presenting an abstract mathematical theory, the Lorenz 1963 system with three ODEs (Eq. 2.1) is used as a working example. The chapter includes an assessment of the SDs computed in such a problem, using the Continuous Adjoint (CA) method [18] and Finite Differences (FD) [18], and the failure to obtain meaningful derivative values using the adjoint method along with the costly success of FD are discussed. LSS is then presented, which is used to successfully compute the SDs.

## 2.1 The Lorenz '63 Problem

The Lorenz 1963 system of ODEs is a well-known example [21, 34, 5] of chaotic systems, comprised of three ODEs in time, and their initial conditions. The system is of ergodic [3] nature, which means that irrespective of the initial conditions used to compute the state variables, long-time averaged functions of the state variables converge to the same value.

This system is a suitable example to demonstrate the failure of adjoint sensitivity analysis methods to compute meaningful derivatives, and the potential of the LSS

algorithm to resolve this issue with acceptable computational cost, contrary to the FD method, which is prohibitively costly even at small mathematical systems. The Lorenz '63 system of equations is given below:

$$\frac{dx}{dt} = \sigma(y - x), \qquad x(0) = x_0 \tag{2.1a}$$

$$\frac{dy}{dt} = x(\rho - z) - y, \qquad y(0) = y_0 \tag{2.1b}$$

$$\frac{dz}{dt} = xy - \beta z, \qquad z(0) = z_0 \tag{2.1c}$$

In this diploma thesis, the parameter $\rho$ is seen as the design variable in a hypothetical optimization problem, in which an objective function (to be developed in a subsequent subsection), cast in the form of a time integral of the state vector $\vec{u}(t,\rho) = [x(t,\rho), y(t,\rho), z(t,\rho)]^T$ and possibly the design variable $\rho$ itself, should be minimized. Quantities $\sigma$ and $\beta$ are constants, equal to 10 and $8/3 \approxeq 2.6667$, respectively.

The behavior of the Lorenz system for values of $\rho$ in the range 0 to 100 is the following [4]:

1. $\rho \in [0, 1]$ : Stable fixed point attractor at $\vec{u} = (0, 0, 0)$.
2. $\rho \in (1, 24.74)$ : Two stable fixed point attractors at $x = y = \sqrt{\beta(\rho - 1)}, z = \rho - 1$.
3. $\rho \in (24.74, 31)$ : Quasi-hyperbolic strange attractors.
4. $\rho \in (31, 99.5)$ : Non-hyperbolic quasi-attractors.
5. $\rho \in [99.5, 100]$ : Periodic limit cycle attractors with an infinite series of period doubling.

Fig. 2.1 shows the solution to the Lorenz '63 equations for some $\rho$ values in the aforementioned intervals, using a Runge Kutta $4^{th}$ order method [17] with $N = 50000$ and $\Delta t = 0.001$ time unit, thus the integration time is $T = N\Delta t = 50$ time units. The initial conditions are $\vec{u}_0 = [x_0, y_0, z_0]^T = [1, 2, 30]^T$.

Fig. 2.2 illustrates the chaotic behavior of the system by comparing the $z$ component of two solutions of the Lorenz '63 problem, for $\rho = 28.00$, and 28.01, obtained using the same $4^{th}$ order Runge-Kutta method. Initially, the two solution trajectories are very close to each other. After that, however, due to the chaotic nature of the Lorenz system, they differ greatly.

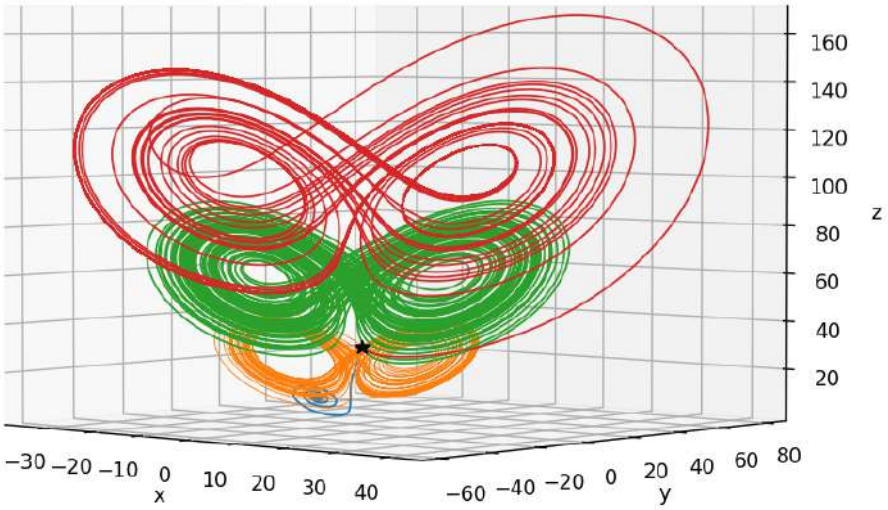



Figure 2.1: *Lorenz '63* Solution for $\rho = 10$ (in blue), $\rho = 28$ (in orange), $\rho = 60$ (in green) and $\rho = 100$ (in red), with the constants and initial conditions described in the text. The initial point (common in all cases) is marked with a black star.

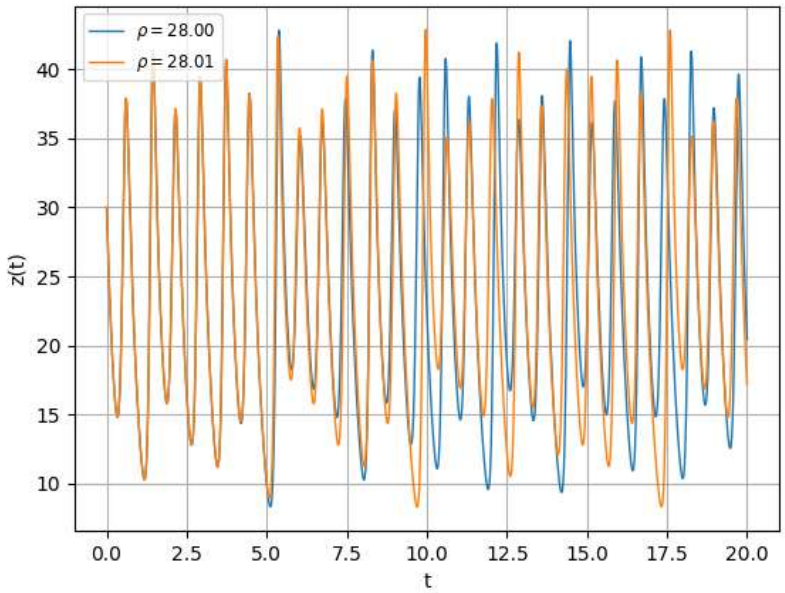



Figure 2.2: *Lorenz '63* Computed time-series $z(t)$ for $\rho = 28.00$ (in blue) and $\rho = 28.01$ (in orange), with the constants and initial conditions described in the text.

## 2.2 Continuous Adjoint Sensitivity Analysis

Throughout this analysis, for the purpose of demonstration, the objective function F is defined as:

$$F(\rho) = \frac{1}{T}\int_0^T z(t,\rho)\,dt \tag{2.2}$$

F is the long-time averaged $z$, and, as already stated, the design parameter vector is $\vec{b} = [b_0]\,, b_0 = \rho$. In order to formulate the CA method and compute the gradient of $F$ w.r.t. $\vec{b}$, the augmented objective function is defined as:

$$F_{aug} = F + \int_0^T \vec{\Psi}^T \vec{R}\,dt \tag{2.3}$$

where $\vec{\Psi} = [\Psi_x, \Psi_y, \Psi_z]^T \in \mathbb{R}^3$ is the adjoint variable vector field and $\vec{R} \in \mathbb{R}^3$ are the residuals of the three Lorenz equations:

$$\vec{R} = \begin{pmatrix} \frac{dx}{dt} - \sigma(y-x) \\ \frac{dy}{dt} - x(\rho - z) + y \\ \frac{dz}{dt} - xy + \beta z \end{pmatrix} \tag{2.4}$$

Eq. 2.3 is then written as:

$$\begin{aligned} F_{aug} = \frac{1}{T}\int_0^T z(t)\,dt + \int_0^T \Psi_x \left(\frac{dx}{dt} - \sigma(y-x)\right)\,dt + \int_0^T \Psi_y \left(\frac{dy}{dt} - x(\rho - z) + y\right)\,dt \\ + \int_0^T \Psi_z \left(\frac{dz}{dt} - xy + \beta z\right)\,dt \end{aligned} \tag{2.5}$$

Differentiating Eq. 2.5 w.r.t. $\rho$ and since the operators $\frac{\delta}{\delta\rho}$ and $\frac{d}{dt}$ are interchangeable, results in:

$$\begin{aligned} \frac{\delta F_a ug}{\delta\rho} = \frac{1}{T}\int_0^T \frac{\delta z}{\delta\rho}\,dt + \int_0^T \Psi_x \frac{d}{dt}\left(\frac{\delta x}{\delta\rho}\right)\,dt + \int_0^T \Psi_y \frac{d}{dt}\left(\frac{\delta y}{\delta\rho}\right)\,dt + \int_0^T \Psi_z \frac{d}{dt}\left(\frac{\delta z}{\delta\rho}\right)\,dt \\ + \int_0^T \left[\Psi_x \sigma\left(\frac{\delta x}{\delta\rho} - \frac{\delta y}{\delta\rho}\right) + \Psi_y \left(z\frac{\delta x}{\delta\rho} + x\frac{\delta z}{\delta\rho} - x - \rho\frac{\delta x}{\delta\rho} + \frac{\delta y}{\delta\rho}\right)\right. \\ \left. + \Psi_z \left(-\frac{\delta x}{\delta\rho}y - \frac{\delta y}{\delta\rho}x + \beta\frac{\delta z}{\delta\rho}\right)\right]\,dt \end{aligned} \tag{2.6}$$

By integrating the time derivative terms by parts, for $u_k = x, y, z$ and $\Psi_k = \Psi_x, \Psi_y, \Psi_z$ respectively:

$$\int_0^T \Psi_k \frac{d}{dt}\left(\frac{\delta u_k}{\delta \rho}\right)\, dt = \left[\Psi_k \frac{\delta u_k}{\delta \rho}\right]_0^T - \int_0^T \frac{d\Psi_k}{dt}\frac{\delta u_k}{\delta \rho}\, dt \tag{2.7}$$

After factoring out terms $\frac{\delta u_k}{\delta \rho}$, Eq. 2.6 results in:

$$\begin{aligned}\frac{\delta F_{aug}}{\delta \rho} = &\int_0^T \frac{\delta x}{\delta \rho}\left(-\frac{d\Psi_x}{dt} + \Psi_x\sigma + \Psi_y\left(z-\rho\right) - \Psi_z y\right)\, dt \\ &+ \int_0^T \frac{\delta y}{\delta \rho}\left(-\frac{d\Psi_y}{dt} - \Psi_x\sigma + \Psi_y - \Psi_z x\right)\, dt \\ &+ \int_0^T \frac{\delta z}{\delta \rho}\left(-\frac{d\Psi_z}{dt} + \Psi_y x + \beta\Psi_z + \frac{1}{T}\right)\, dt \\ &+ \left[\Psi_x \frac{\delta x}{\delta \rho}\right]_0^T + \left[\Psi_y \frac{\delta y}{\delta \rho}\right]_0^T + \left[\Psi_z \frac{\delta z}{\delta \rho}\right]_0^T - \int_0^T \Psi_y x\, dt\end{aligned} \tag{2.8}$$

The Field Adjoint Equations (FAEs) are determined by setting the multipliers of $\frac{\delta u_k}{\delta \rho}$ to zero continuously throughout the interval $0 \leq t < T$. The Adjoint Boundary Conditions (ABCs) are determined by setting the coefficients of terms $\frac{\delta u_k}{\delta \rho}$ to zero at $t = T$. The FAEs and ABCs are:

$$\frac{d\Psi_x}{dt} = \Psi_x\sigma + \Psi_y\left(z-\rho\right) - \Psi_z y \tag{2.9a}$$

$$\frac{d\Psi_y}{dt} = -\Psi_x\sigma + \Psi_y - \Psi_z x \tag{2.9b}$$

$$\frac{d\Psi_z}{dt} = \Psi_y x + \beta\Psi_z + \frac{1}{T} \tag{2.9c}$$

$$\Psi_x(T) = \Psi_y(T) = \Psi_z(T) = 0 \tag{2.9d}$$

which imply that, as is always the case in unsteady adjoint, the adjoint equations should be integrated backward in time. In vector notation, Eq. 2.9 is written as:

$$\frac{d\vec{\Psi}}{dt} = \begin{pmatrix} \sigma & (z-\rho) & -y \\ 1 & -\sigma & -x \\ 0 & x & \beta \end{pmatrix}\vec{\Psi} + \begin{pmatrix} 0 \\ 0 \\ \frac{1}{T} \end{pmatrix}, \qquad \vec{\Psi}\left(T\right) = \vec{0} \tag{2.10}$$

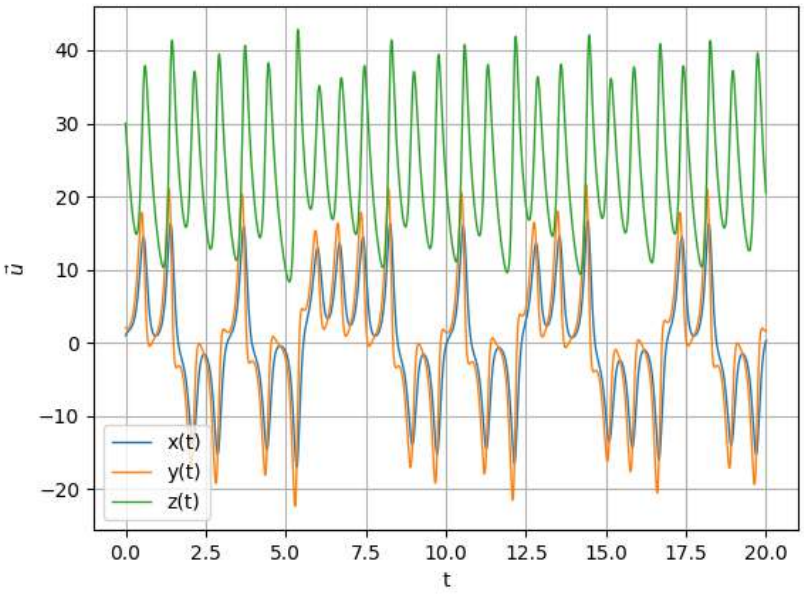



Figure 2.3: *Lorenz '63* The solution $\vec{u} = [x, y, z]^T$ for $\rho = 28$, T = 20 time units, and the parametric values mentioned in the text.

After eliminating the terms $\frac{\delta u_k}{\delta \rho}$ by satisfying the FAEs and ABCs, the only remaining term on the right-hand-side of Eq. 2.8 constitutes the SD:

$$\frac{\delta F}{\delta \rho} = -\int_0^T \Psi_y x \, dt \tag{2.11}$$

To solve the FAEs, the same $4^{th}$ order Runge-Kutta method is used to integrate Eqs. 2.10 backward in time with, of course, the same time step $\Delta t = 0.001$. After having computed the adjoint fields, the SD results from the integral of Eq. 2.11. The objective function for $\rho = 28$, $T = 20$ and the parametric values mentioned in the text, is $F = 23.34$, and the corresponding SD is $\frac{\delta F}{\delta \rho} = 341650.3$, which absolutely wrong compared to the known correct value, namely 1.01 [32].

The adjoint vector fields plotted in Fig. 2.4, and the corresponding primal solution $\vec{u}(t)$ in Fig. 2.3. Also, in Fig. 2.5, the time series of $\Psi_y(t), x(t)$ and the negative of their product $-\Psi_y(t) \cdot x(t)$, the integral of which constitutes the SD, are plotted against $t$. Fig. 2.6 shows the cumulative integral of Eq. 2.11 $(-x \cdot \Psi_y)$ as it is integrated forward from 0 to $t'$ (blue curve) and backward from $T$ to $t'$ (red curve), where $0 \leq t' \leq T$.

The value of the objective function $F$ and the absolute value of the SD $\frac{\delta F}{\delta \rho}$ was plotted against different integration times $T$, in Figs. 2.7 and 2.8 respectively, for

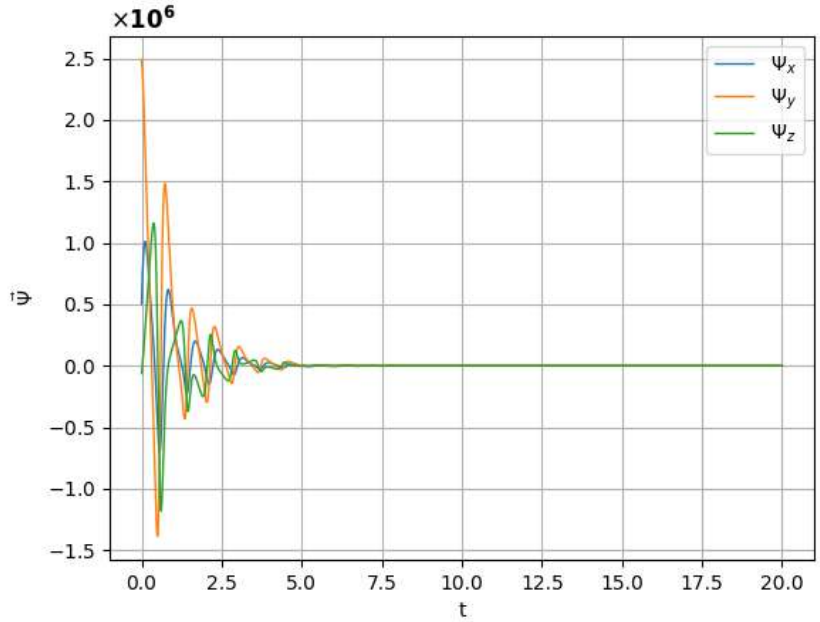



Figure 2.4: *Lorenz '63* The adjoint vector field components $\Psi_x, \Psi_y$ and $\Psi_z$ vs. $t$, for $\rho = 28$, T = 20 time units, and the parametric values mentioned in the text. The three curves were integrated from right to left (backward in time).

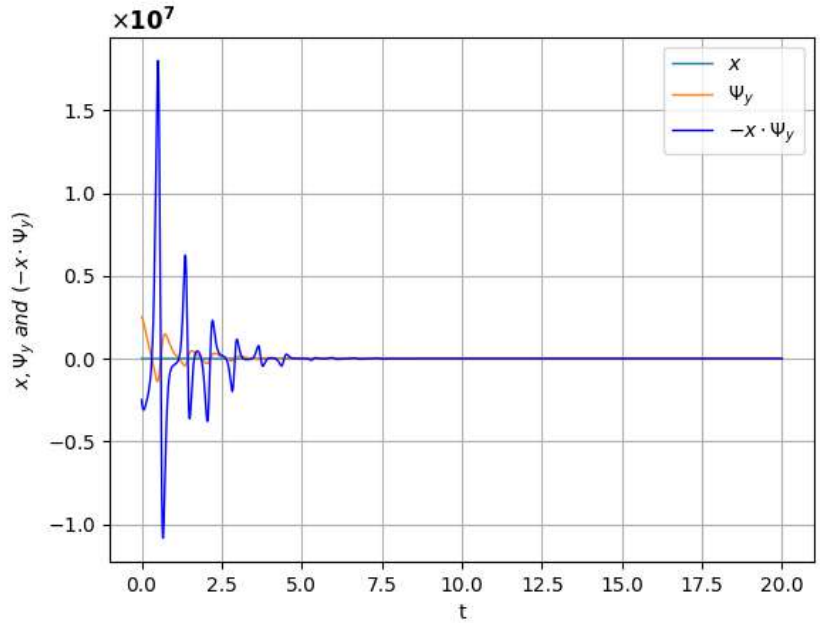



Figure 2.5: *Lorenz '63* Time-series of $x$, $\Psi_y$ and their negative product $(-x \cdot \Psi_y)$ for $\rho = 28$, T = 20 time units, and the parametric values mentioned in the text.

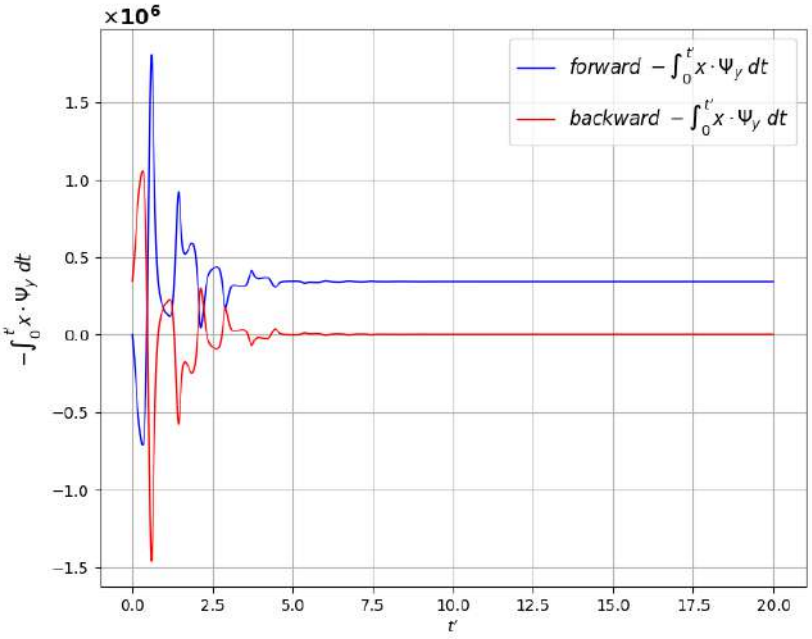



Figure 2.6: *Lorenz '63* Time series $(-x \cdot \Psi_y)$ integrated forward from 0 to $t'$ and backward from $T$ to $t'$, where $0 \leq t' \leq T$, for $\rho = 28$, T = 20 time units, and the parametric values mentioned in the text. In both cases, regardless of the direction of integration, the SD is $\frac{\delta F}{\delta \rho} = 341650.3$.

$\rho = 28$. The values of $F$ do not converge even at very long integration times, but vary between 23.50 and 23.55. Onm the other hand, the SD cannot converge to a single value, and it increases up to the limit of what double precision can represent, for increasing $T$. The absolute values of the SDs were chosen instead of the signed values, because the latter were both positive and negative, and the negative values are undefined on the logarithmic scale.

The value of the objective function $F$ and the SD $\frac{\delta F}{\delta \rho}$ can be seen in Figs. 2.9 and 2.10 respectively. For each $\rho$ value, both have been computed for 20 random initial conditions. The increase of F appears to be linear, with a slope of about 1.0, with close to no variation for different initial conditions, as it can be seen by the vertical dots being practically indistinguishable from one another. That is expected because the Lorenz '63 system is ergodic. However, the SD values are meaningless and do not convey the aforementioned linearity, by being approximately 1.01, but instead they diverge.

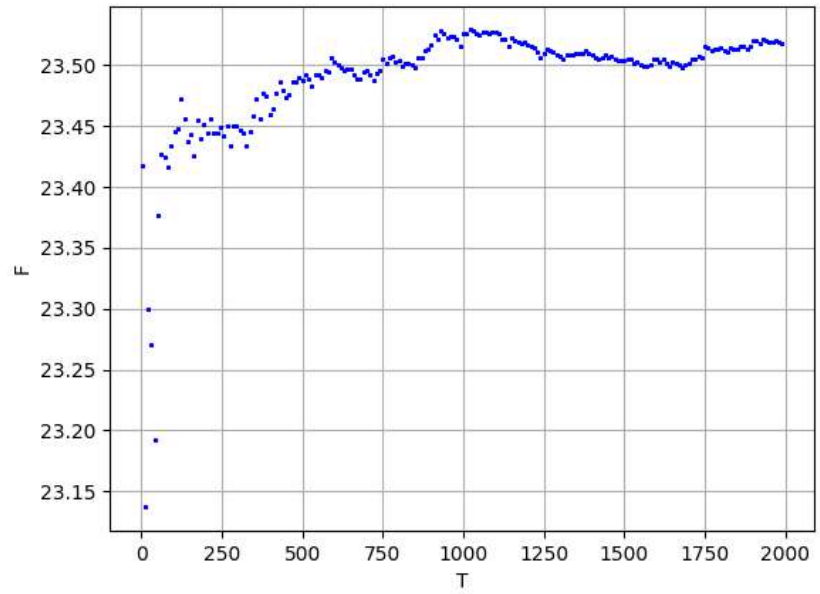



Figure 2.7: *Lorenz '63* Objective function F values for integration times $T$ from 1 to 2000, for $\rho = 28$.

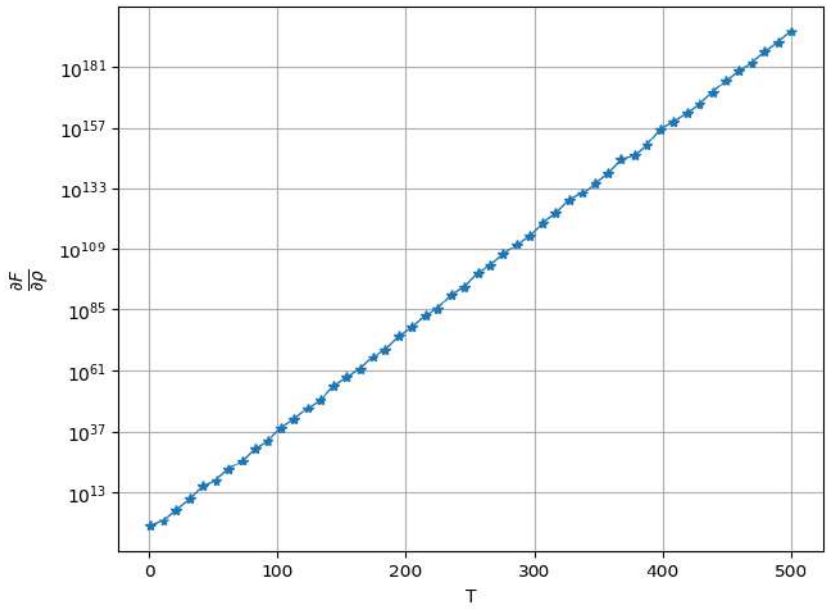



Figure 2.8: *Lorenz '63* SD $\frac{\delta F}{\delta \rho}$ values computed using the CA method, for integration times $T$ from 1 to 500, for $\rho = 28$.

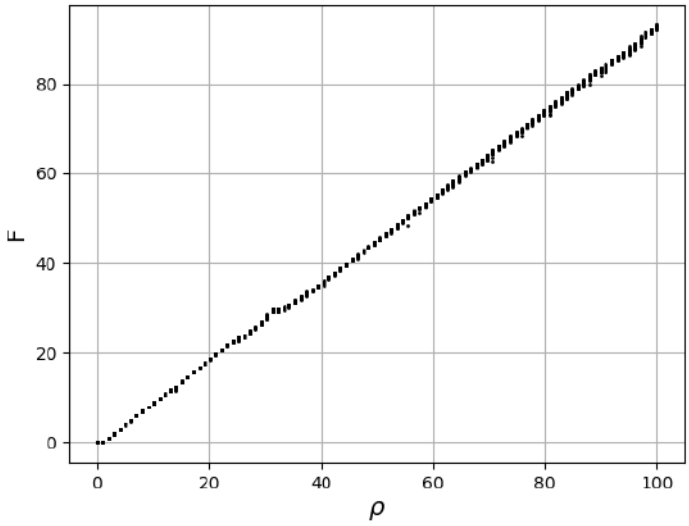



Figure 2.9: *Lorenz '63* Objective function F values for $\rho$ from 0 to 100, for 20 random initial conditions.

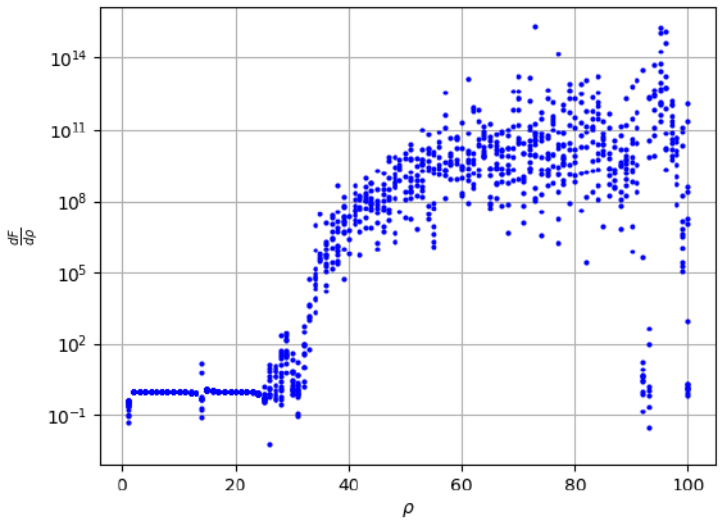



Figure 2.10: *Lorenz '63* SD values computed using the CA method, for $\rho$ from 0 to 100, for 20 random initial conditions.

### 2.2.1 Parametric Study on the Step Size $\Delta t$

Three of $\Delta t$ were tested in order to determine the grid-size dependence of the primal and CA solutions of the Lorenz '63 system. Figs. 2.11 and 2.12 show the $x(t)$ time series derived from the solution of Eq. 2.1 and the $\Psi_y(t)$ time series derived from the solution of Eq. 2.9 for $\Delta t = 10^{-3}, 10^{-4}$ and $10^{-5}$, and $\rho = 28$, $T = 20$ time units. The trajectories are practically indistinguishable, meaning that the selection of $\Delta t = 10^{-3}$ did not affect the quality of the results.

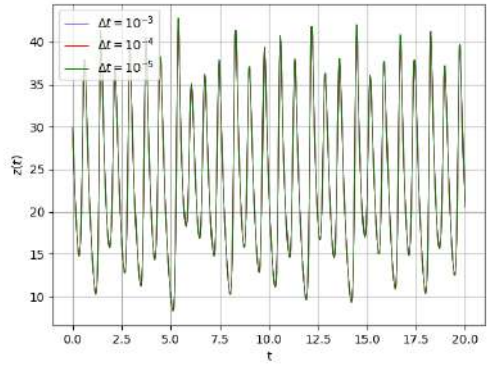



Figure 2.11: *Lorenz '63* $x(t)$ for $\Delta t = 10^{-3}, 10^{-4}$ *and* $10^{-5}$, $\rho = 28$ and $T = 20$.

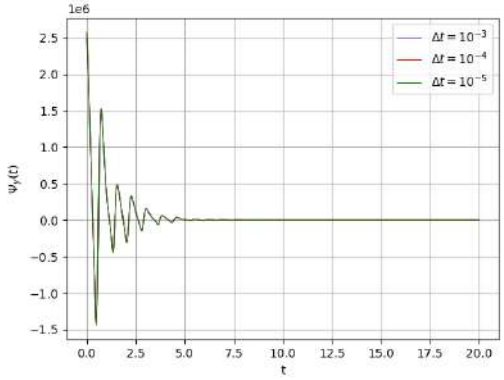



Figure 2.12: *Lorenz '63* $\Psi_y(t)$ for $\Delta t = 10^{-3}, 10^{-4}$ *and* $10^{-5}$, $\rho = 28$ and $T = 20$.

## 2.3 Finite Differences Sensitivity Analysis

The second method used to compute the SD of Eq. 2.2 is the FD method. The same range was used for $\rho$, from 0 to 100, with step equal to $\Delta\rho = 2$. The perturbation or step of the FD appears to be unexpectedly high but this will be discussed later on. The formula for the second-order, central FD method is:

$$\frac{\delta F}{\delta \rho} = \frac{F(\rho + \Delta\rho) - F(\rho - \Delta\rho)}{2\Delta\rho} \tag{2.12}$$

In Fig. 2.13, the SD for $\rho = 28$ was evaluated for different integration times T ranging from 1 to 2000 time units, with $\Delta t = 0.001$ time units. It should be noted that the SDs computed are meaningful, unlike the CA method (Fig. 2.10), and are in fact close to the correct value of around 1.01, and seem to converge to that value for integration times above $T = 1800$ time units. As was mentioned previously, however, integration times that long are extremely costly and impractical for real-world problems.

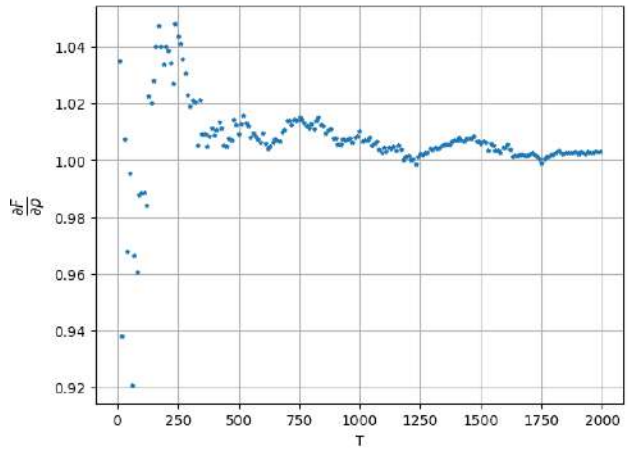



Figure 2.13: *Lorenz '63* SD computed using FD, for values of integration time T ranging from 1 to 2000 time units, for $\rho = 28$, $\Delta\rho = 2$, and the same initial conditions.

The effect of the perturbation $\Delta\rho$ used in Eq. 2.12 on the SD value can be seen in Fig. 2.14, where the SD was plotted against corresponding values of $\Delta\rho$ ranging from $10^{-6}$ to 2.0, for $\rho = 28$. It is shown that, for values of $\Delta\rho$ between 1.0 and 2.0, the SD value is more or less stable, and close to the reference value of 1.01 [32]. As a result, the perturbation was chosen to be $\Delta\rho = 2$. Figs. 2.15 and 2.16 show, in more detail, the SDs at $10^{-6} \leq \Delta\rho \leq 10^{-2}$ (Fig. 2.15) and $0.5 \leq \Delta\rho \leq 2.0$ (Fig. 2.16), also for $\rho = 28$.

In Fig. 2.17 the SD is computed using FD for integration time $T = 20$ time units(in blue) and $T = 2000$ time units (in red), for $0 \leq \rho \leq 100$. At each $\rho$ value, the SD for 20 random initial conditions was plotted. The SD values of the larger integration time are obviously more accurate and with minimal deviation, compared to the smaller integration time, and they are both very close to the previously known correct value of approximately 1.01 [32]. However, the FD method is computationally expensive as it requires a very long integration time to compute useful derivatives. It should be noted that in more complex real applications, namely CFD simulations, such integration times are prohibitive.

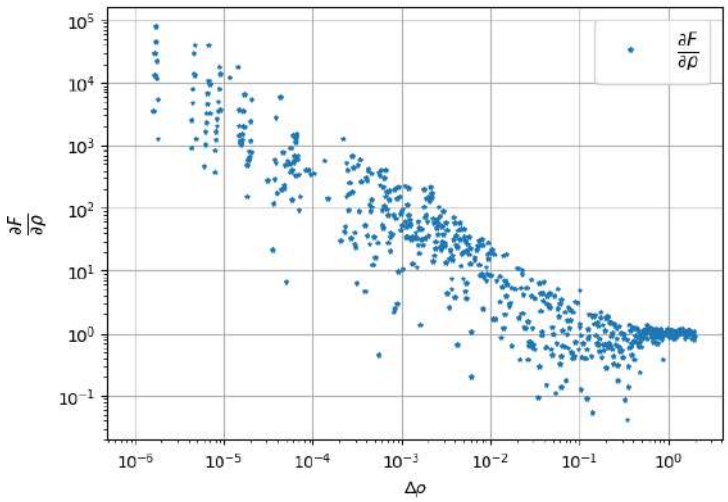



Figure 2.14: *Lorenz '63* SD vs. the corresponding perturbation $\Delta\rho$ used in the FD method, for $\rho = 28$ and $T = 20$.

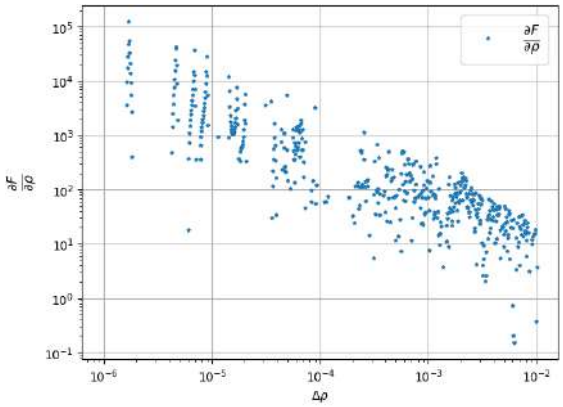



Figure 2.15: *Lorenz '63* SD for $10^{-6} \leq \Delta\rho \leq 10^{-2}$.

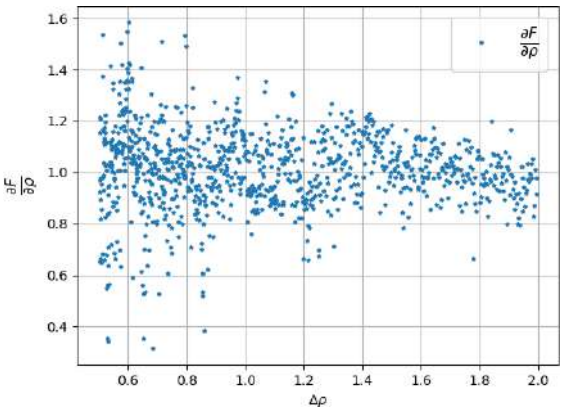



Figure 2.16: *Lorenz '63* SD for $0.5 \leq \Delta\rho \leq 2.0$.

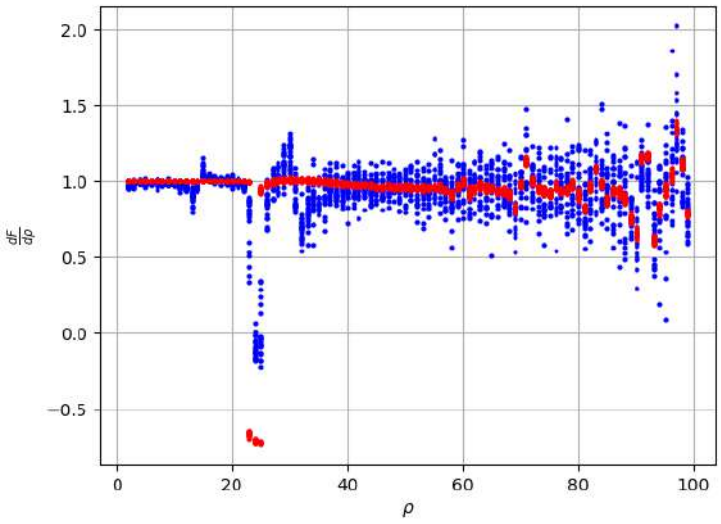



Figure 2.17: *Lorenz '63* SD (vertical axis), using FD with step $\Delta\rho = 2$, for $\rho$ values ranging from 1 to 100 (horizontal axis). The blue points correspond to integration time $T = 20$ and the red ones to $T = 2000$.

## 2.4 Least Squares Shadowing (LSS) Sensitivity Analysis

The Least Squares Shadowing (LSS) algorithm [5, 21, 33, 9, 34, 26, 4, 13, 10, 7, 3, 32] aims at overcoming the difficulties faced by Adjoint methods and produce meaningful SDs of long time averaged quantities w.r.t. a design parameter. It requires the basic assumption of *ergodicity* of the system, meaning that the long time behavior of the system is independent of the initial conditions. This assumption enables the "comparison" of two trajectories, one with an unperturbed ($\rho$) and another with a perturbed ($\rho+\delta\rho$) parameter value, with different initial conditions. In this context, a trajectory refers to $\vec{u}(t)$, for $0 \leq t \leq T$, where $\vec{u}(t) = [x(t), y(t), z(t)]^T$, computed for some value of $\rho$. The perturbed parameter trajectory, computed for $\rho + \Delta\rho$, is its "shadow trajectory".

This way, a different initial condition and a time transformation $\tau(t)$ can be chosen for the shadow trajectory, so that the two trajectories remain close to each other throughout $0 \leq t \leq T$. The reference trajectory $\vec{u}_r$ for design variable value $\rho$ and the shadow trajectory $\vec{u}$ for $\rho + \delta\rho$, as well as $t$ and $\tau(t)$, need to be very close to each other in order for the LSS method to compute a meaningful SD.

The existence of a shadow trajectory $\vec{u}$ and its convergence to the reference trajectory $\vec{u}_r$ for $\delta\rho \to 0$ is guarantied by the *shadowing lemma* [3], which states:

### 2.4.1 Shadowing Lemma [3]

*Consider a reference solution $\vec{u}_r$ to*

$$\frac{d\vec{u}_r}{dt} = \vec{f}(\vec{u}_r, \rho) \tag{2.13}$$

*If this system has a hyperbolic strange attractor and if some system parameter $\rho$ is slightly perturbed by some $\Delta\rho$: For any $\delta > 0$ there exists $\epsilon > 0$, such that for every $\vec{u}_r$ that satisfies $\|d\vec{u}_r/dt - \vec{f}(\vec{u}_r, \rho)\| < \epsilon$, there exists a true solution $\vec{u}$ and a time transformation $\tau(t)$, such that $\|\vec{u}(\tau(t)) - \vec{u}_r(t)\| < \delta, |1 - d\tau/dt| < \delta$ and $d\vec{u}/d\tau - \vec{f}(\vec{u}, \rho+\Delta\rho) = 0$, where $\|\cdot\|$ is the L2 norm in phase space. The trajectories $\vec{u}$ and $\vec{u}_r$ symbolize the perturbed and unperturbed trajectories respectfully.*

### 2.4.2 Computing the Shadow Trajectory as a Minimization Problem

In order to find the shadow trajectory $\vec{u}(t)$ and a time transformation $\tau(t)$ so the reference and the shadow trajectory remain close to each other, a constrained minimization problem is solved:

$$\min_{\vec{u},\eta} \frac{1}{2}\int_0^T \|\vec{u}\,(\tau(t))-\vec{u}_r\|^2+\alpha^2\left(1-\frac{d\tau}{dt}\right)^2 dt, \qquad s.t. \quad \frac{d\vec{u}}{d\tau}=\vec{f}(\vec{u},\rho+\delta\rho), \qquad 0<t<T \tag{2.14}$$

where $\vec{f}(\vec{u},\rho)$ denotes the right-hand-sides of Eq. 2.1:

$$\vec{f}(\vec{u},\rho)=[f_1(\vec{u},\rho),f_2(\vec{u},\rho),f_3(\vec{u},\rho)]^T=[\sigma(y-x),x(\rho-z)-y,xy-\beta z]^T \tag{2.15}$$

Eq. 2.14 ensures that the two trajectories $\vec{u}(t)$ and $\vec{u}_r(t)$, as well as $\tau(t)$ and $t$, remain as close to each other as possible for all $0 \leq t \leq T$. The time transformation is defined as:

$$\tau(t)=(1+\eta(t)\,\delta\rho)t, \quad \eta(0)=0 \tag{2.16}$$

which is also guarantied to be close to $t$ by Eq. 2.14. Parameter $\alpha$ is a weighting parameter chosen so that the terms $\|\vec{u}-\vec{u}_r\|^2$ and $\alpha^2\left(1-\frac{d\tau}{dt}\right)^2$ are of the same order of magnitude [13]. In the case of the Lorenz '63 equations, $\alpha=30$.

In order to solve Eq. 2.14, this is processed, as in Appendix A.1. The resulting minimization problem is:

$$\min_{\vec{v},\eta} \frac{1}{2}\int_0^T \|\vec{v}\|^2+\alpha^2\eta^2\,dt, \qquad s.t. \quad \frac{d\vec{v}}{dt}=\frac{\partial\vec{f}}{\partial\vec{u}}\vec{v}+\frac{\partial\vec{f}}{\partial\rho}+\eta\vec{f}, \qquad 0<t<T \tag{2.17}$$

or in Einstein notation:

$$\min_{\vec{v},\eta} \frac{1}{2}\int_0^T v_i^2+\alpha^2\eta^2\,dt, \qquad s.t. \quad \frac{dv_i}{dt}=\frac{\partial f_i}{\partial u_j}v_j+\frac{\partial f_i}{\partial\rho}+\eta f_i, \qquad 0<t<T \tag{2.18}$$

where $v_i \equiv \frac{\partial u_i}{\partial\rho}$. The constraints in Eqs. 2.17 and 2.18 correspond to the Direct Differentiation (DD; a.k.a. Tangent) equation, which functions as the primal equation in the above optimization problem.

The Karush-Kuhn-Tucker (KKT) equations that correspond to the minimization

problem of Eq. 2.18 are derived in Appendix A.2, and in Einstein notation are:

$$\frac{dv_i}{dt} = \frac{\partial f_i}{\partial u_j} v_j + \frac{\partial f_i}{\partial \rho} + \eta f_i \tag{2.19a}$$

$$\frac{dw_i}{dt} = -\frac{\partial f_j}{\partial u_i} w_j + v_i \tag{2.19b}$$

$$\eta = \frac{1}{\alpha^2} w_i f_i \tag{2.19c}$$

$$w_i(0) = w_i(T) = 0 \tag{2.19d}$$

The equivalent of Eq. 2.19 in vector notation is:

$$\frac{d\vec{v}}{dt} = \frac{\partial \vec{f}}{\partial \vec{u}} \vec{v} + \frac{\partial \vec{f}}{\partial \rho} + \eta \vec{f} \tag{2.20a}$$

$$\frac{d\vec{w}}{dt} = -\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T \vec{w} + \vec{v} \tag{2.20b}$$

$$\eta = \frac{1}{\alpha^2} \vec{w}^T \vec{f} \tag{2.20c}$$

$$\vec{w}(0) = \vec{w}(T) = 0 \tag{2.20d}$$

The problem consists of a system of three equations and an equal number of unknowns, $\vec{u} \in \mathbb{R}^3$ (3 by 1 vector), $\vec{v} \in \mathbb{R}^3$ and $\eta \in \mathbb{R}$.

It should be noted that the KKT equations admit two Boundary Conditions (BCs) for $\vec{w}$, determining its first and last values, namely $\vec{w}(0) = \vec{w}(T) = 0$. However, Eq. 2.20b is a first degree ODE of $\vec{w}$ and Eq. 2.20a is also a first degree ODE of $\vec{v}$ and no BC is directly defined for $\vec{v}$. In order to resolve this issue, one could use a shooting method and impose a BC for $\vec{v}$ at $t = 0$, the value of which would be computed with trial and error until $\vec{w} = 0$ at $t = T$. To avoid this impractical procedure, an idea proposed in [4] is to transform the initial value problem into a boundary value problem in time, in terms of $\vec{w}$, which easily accommodates the two BCs. Thus, the KKT system becomes well-defined. In order to solve Eqs. 2.20, they are be combined into a single second order ODE of $\vec{w}$, as in Appendix A.2:

$$\begin{gathered}\frac{d^2\vec{w}}{dt^2} + \left[\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T - \frac{\partial \vec{f}}{\partial \vec{u}}\right] \frac{d\vec{w}}{dt} + \left[\frac{d}{dt}\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T - \frac{\partial \vec{f}}{\partial \vec{u}}\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T - \frac{1}{a^2}\vec{f}\vec{f}^T\right] \vec{w} - \frac{\partial \vec{f}}{\partial \rho} = 0, \\ \vec{w}(0) = \vec{w}(T) = 0\end{gathered} \tag{2.21}$$

This is a second-order ODE which corresponds to a boundary value problem in which, the BCs can be properly imposed. As shown in Appendix A.2, Eq. 2.21 can

be simplified to:

$$\frac{d^2\vec{w}}{dt^2} + \mathbf{A}\frac{d\vec{w}}{dt} + \mathbf{B}\vec{w} - \mathbf{C} = \vec{0}, \qquad \vec{w}|_{t=0} = \vec{w}|_{t=T} = 0$$
$$\text{where: } \mathbf{A} = \frac{\partial \vec{f}}{\partial \vec{u}} - \left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T, \qquad \mathbf{B} = \frac{d}{dt}\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T - \frac{\partial \vec{f}}{\partial \vec{u}}\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T - \frac{1}{\alpha^2}\vec{f}\vec{f}^T, \qquad \vec{C} = \frac{\partial \vec{f}}{\partial \rho} \tag{2.22}$$

The simplified expressions of the coefficients $\mathbf{A}$, $\mathbf{B}$ and $\vec{C}$ for the Lorenz '63 system are also found in Appendix A.2.

In order for Eq. 2.22 to be solved, it is discretized using $2^{nd}$ order central FD, for $N$ timesteps with interval size $\Delta t$. For nodes $0 \leq i \leq N-1$:

$$\begin{aligned}
&\vec{R}_i = \vec{w}_i - 0, \qquad for\ i = 0 \\
&\vec{R}_i = \frac{\vec{w}_{i-1} - 2\vec{w}_i + \vec{w}_{i+1}}{\Delta t^2} + \mathbf{A}_i\frac{-0.5\vec{w}_{i-1} + 0.5\vec{w}_{i+1}}{\Delta t} + \mathbf{B}_i\vec{w}_i - \vec{C}_i, \quad for\ 1 \leq i \leq N-2 \\
&\vec{R}_i = \vec{w}_i - 0, \qquad for\ i = N-1
\end{aligned} \tag{2.23}$$

which can be written as:

$$\begin{aligned}
&\vec{R}_i = \vec{w}_i, \qquad for\ i = 0 \\
&\vec{R}_i = \left(I - 0.5\Delta t\mathbf{A}_i\right)\vec{w}_{i-1} + \left(-2I + \Delta t^2\mathbf{B}_i\right)\vec{w}_{i-1} + \left(I + 0.5\Delta t\mathbf{A}_i\right)\vec{w}_{i+1} - \Delta t^2\vec{C}_i, \quad for\ 1 \leq i \leq N-2 \\
&\vec{R}_i = \vec{w}_i, \qquad for\ i = N-1
\end{aligned} \tag{2.24}$$

where $\mathbf{I}$ is the 3 by 3 unit matrix.

To solve the discretized system of equation 2.24 the Block TriDiagonal Matrix Algorithm (BTDMA) is employed. The system can be written in the following form:

$$\mathbf{M}\mathbf{w} = \mathbf{G} \tag{2.25}$$

where $\mathbf{M}$ is s a $N$ by $N$ tridiagonal block matrix, with block size 3 by 3, $\mathbf{w}$ is a $N \times 3$ matrix that holds the $\vec{w}$ values for each time-step, and $\mathbf{G}$ is the right-hand-side term of the system equation, and is also $N \times 3$, with each row being $\mathbf{G}_i = \Delta t^2\vec{C}_i$. Matrix $\mathbf{M}$ contains the following elements:

$$\begin{cases}
\mathbf{M}_{i,j} = I, \qquad for\ i = 0\ and\ i = N-1 \\
\mathbf{M}_{i,i-1} = I - 0.5\Delta t\mathbf{A}_i, \quad \mathbf{M}_{i,i} = -2I + \Delta t^2\mathbf{B}_i, \quad \mathbf{M}_{i,i+1} = I + 0.5\Delta t\mathbf{A}_i, \qquad for\ 1 \leq i \leq N-2
\end{cases} \tag{2.26}$$

### 2.4.3 Evaluation of the Sensitivity Derivative

After solving Eq. 2.25, time-series $\vec{v}$ and $\eta$ can be computed for each node, as in Appendix A.3. Afterwards, the following equation is used to evaluate the SD.

$$\frac{\delta F}{\delta \rho} = \frac{1}{T}\int_0^T (v_z + \eta z)\ dt - \frac{1}{T^2}\int_0^T \eta\, dt \int_0^T z\, dt \tag{2.27}$$

Eq. 2.27 is derived in Appendix A.4.

### 2.4.4 The LSS Flowchart

The implementation of the LSS method for the computation of the SD $\frac{\delta F}{\delta \rho}$ can be broken down into the following steps:

1. Solve the Lorenz equations (Eq. 2.1) using a forward-in-time, fourth order Runge-Kutta scheme, and store the $\frac{\partial \vec{f}}{\partial \vec{u}}$ and $\frac{\partial \vec{f}}{\partial \rho}$ time-series.
2. Solve the boundary value problem of Eq. 2.22 using the BTDMA algorithm, as in Appendix A2, and store $\vec{w}(t)$.
3. Evaluate the SD using Eq. 2.27, without needing to store the $\vec{v}(t)$ and $\eta(t)$ time-series, while computing $\vec{v}(t)$ and $\eta(t)$ as in Appendix A.3.

### 2.4.5 Results from the LSS Method

Fig. 2.18 shows the reference and shadow trajectories $\vec{u}_r$ and $\vec{u}$ with blue and red color respectively, for integration time $T = 15$ time units, $\rho = 28$ and $\alpha = 30$. It can be observed that they remain close to each another for the entire time as intended, regardless of the chaotic nature of the Lorenz '63 system. Fig. 2.19 shows component $z(t)$ for the reference and shadow trajectories in blue and black respectively, which is in stark contrast with Fig. 2.2 which shows the perturbed and unperturbed trajectories for $\rho = 28.01$ and $\rho = 28.00$ respectively, that were not computed using the LSS method.

The left axis of Fig. 2.21 shows the SD in black, and the right axis shows the ratio $\frac{\|\vec{u}-\vec{u}_r\|^2}{\alpha^2\left(1-\frac{d\tau}{dt}\right)^2}$ in red, for $\alpha$ ranging from 0 to 2000, for $\rho = 28$, $T = 20$ and 20 random initial conditions. It can be observed that the SD fluctuates less for $40 \leq \alpha \leq 100$, for detail see Fig. 2.22, and it is approximately $1.01 \pm 0.02$, in accordance to previous findings. In this area of $\alpha$ values, the fraction is approximately 1.0, which means that the numerator and the denominator are of the same order of magnitude, as required by the LSS theoretical foundation [4]. This suggests that a SD value produced by LSS is more accurate for ratios closer to 1.0.

Fig. 2.23 shows the SD values produced using LSS for $5 \leq T \leq 350$, for $\alpha = 30$ and $\rho = 28$. The SD is less accurate for $5 \leq T \leq 15$ due to transient effects, as can be validated by Fig. 2.24 that shows the Lorenz '63 solution for integration time $0 \leq T \leq 15$ (blue part) and $15 \leq T \leq 35$ (red part). The blue part of the curve has clearly not reached the right-side attractor, unlike the red part that has entered the alternating phase between the two attractors every one round, as is typical for the Lorenz '63 system at $\rho = 28$. However, for $T \geq 20$, the SD is within the interval $1.01 \pm 0.01$, which is far more accurate than the corresponding CA and FD cases, see Figs. 2.8 and 2.13 respectively. Similar precision was achieved using FD for integration time $T \geq 400$ (20 times longer than LSS), which is impractically expensive, as previously emphasized in the text.

Fig. 2.20 shows SD values computed using the LSS method for $\rho$ between 0 and 100 using 20 random initial conditions for each $\rho$ value, for integration time $T = 20$ time units (blue), and $T = 2000$ time units (red), and $\alpha = 30$. It can be deduced that the SD values for $T = 20$ are far more precise than those produced at the same integration times at Figs. 2.10 and 2.17, for the CA and FD methods respectively. It should be noted that the quality of SD values is better for higher integration times $T$.

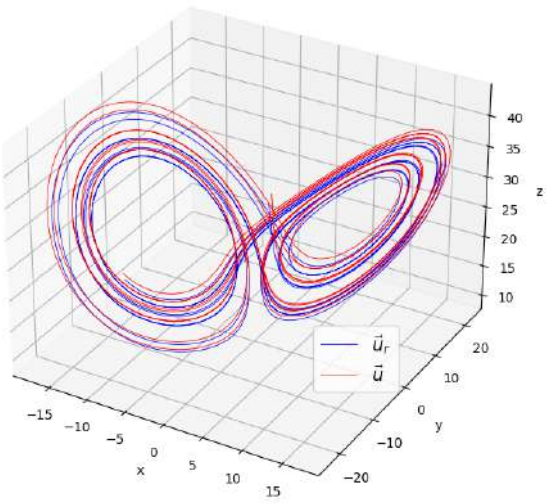



Figure 2.18: *Lorenz '63* The reference and shadow trajectories in blue and red respectively, for $\rho = 28$, integration time $T = 15$ and $\alpha = 30$.

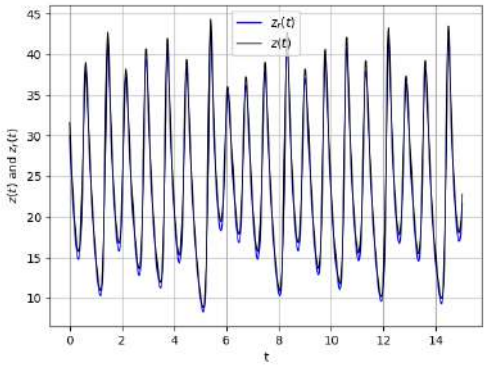



Figure 2.19: *Lorenz '63* Components $z_r(t)$ and $z(t)$ of the reference and shadow trajectories in blue and black color respectively, for $\rho = 28$, integration time $T = 15$, and $\alpha = 30$

### 2.4.6 Discretely Consistent LSS (DCLSS)

A Discretely Consistent version of the LSS algorithm (DCLSS) was developed to ensure the consistency of term $\frac{d\vec{w}}{dt}$ originating from Eq. 2.20b and the time-differentiated

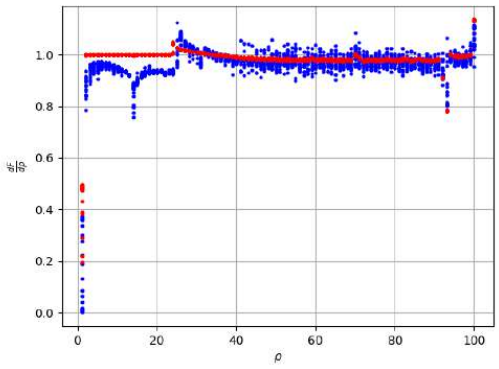



Figure 2.20: *Lorenz '63* The SD values for $\rho$ between 0 and 100, with 20 random initial conditions for each $\rho$ value, computed using the LSS method. The blue points correspond to integration time $T = 20$, and the red ones to $T = 2000$, and $\alpha = 30$

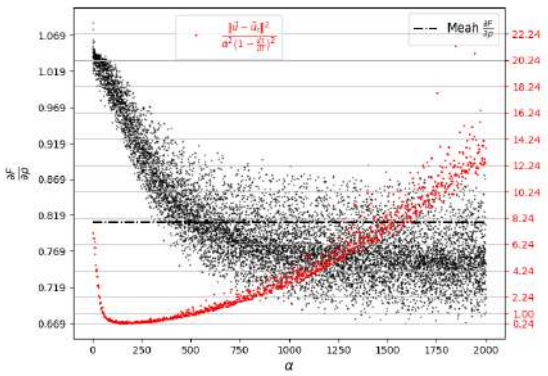



Figure 2.21: *Lorenz '63* Values of SD in black, and the ratio $\frac{\|\vec{u}-\vec{u}_r\|^2}{\alpha^2\left(1-\frac{d\tau}{dt}\right)^2}$ in red, for $\alpha$ ranging from 0 to 2000 and for $\rho = 28$, $T = 20$ and 20 random initial conditions, computed using the LSS method.

Eq. 2.20a. It was derived in order to be used for the tests of the next subsection. In classic LSS, when deriving Eq. A.18, the coefficients of the terms $\frac{d\vec{w}}{dt}$ were factored out, thus making the assumption that $\frac{d\vec{w}}{dt}$ from Eq. 2.20b is identical to the one from Eq. 2.20a. However, in DCLSS Eq. 2.20a was discretized using backward FD and Eq. 2.20b using forward FD. So, from a discrete point of view, the terms $\frac{d\vec{w}}{dt}$ are not identical, and the resulting $2^{nd}$ order ODE is different. The DCLSS algorithm is presented in Appendix A.5, denoting forward FD using a right-pointed arrow and backward FD using a left-pointed arrow. Central FD is used only for the discretization of the $2^{nd}$ order term $\frac{d^2\vec{w}}{dt^2}$ and is not denoted with an arrow. The system and the corresponding $2^{nd}$ order ODE are:

$$\frac{\overleftarrow{d\vec{v}}}{dt} = \frac{\partial \vec{f}}{\partial \vec{u}}\vec{v} + \frac{\partial \vec{f}}{\partial \rho} + \frac{1}{\alpha^2}\vec{f}\vec{f}^T\vec{w} \tag{2.28a}$$

$$\frac{\overrightarrow{d\vec{w}}}{dt} = -\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T \vec{w} + \vec{v} \tag{2.28b}$$

$$\vec{w}(0) = \vec{w}(T) = 0 \tag{2.28c}$$

$$\frac{d^2\vec{w}}{dt^2} + \mathbf{A_1}\frac{\overleftarrow{d\vec{w}}}{dt} - \mathbf{A_2}\frac{\overrightarrow{d\vec{w}}}{dt} + \mathbf{B}\vec{w} = \vec{C} \tag{2.29}$$

where the coefficients are shown in Appendix A.5. The DCLSS trajectories of $\vec{v}$ and $\vec{w}$ are compared with the respective LSS ones on Figs. 2.25 and 2.26. It should be

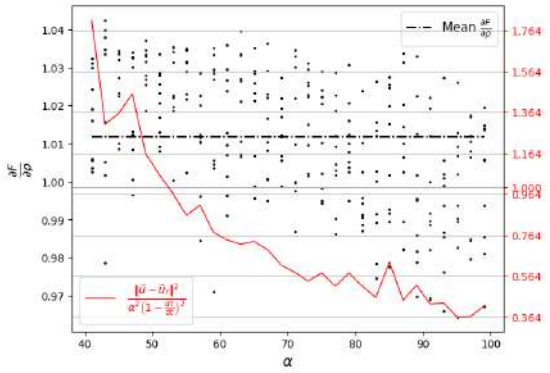



Figure 2.22: *Lorenz '63* Values of SD in black, and the ratio $\frac{\|\vec{u}-\vec{u}_r\|^2}{\alpha^2\left(1-\frac{d\tau}{dt}\right)^2}$ in red, for $\alpha$ ranging from 40 to 100 and for $\rho = 28$, $T = 20$ and 20 random initial conditions, computed using the LSS method.

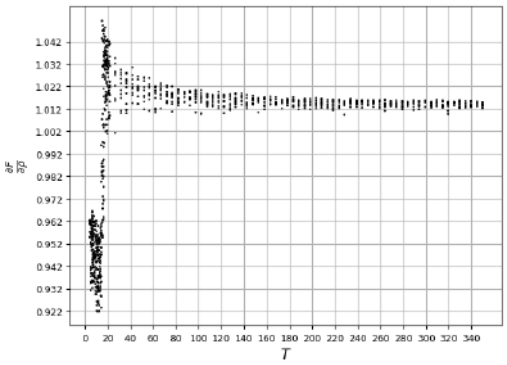



Figure 2.23: *Lorenz '63* SD values for $5 \leq T \leq 350$, for $\alpha = 30$ and $\rho = 28$, computed using the LSS method.

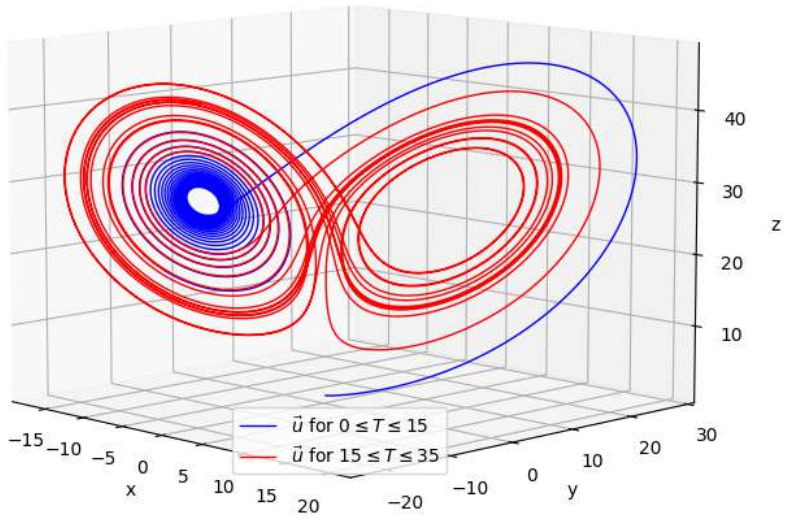



Figure 2.24: *Lorenz '63* Solution for integration time $0 \leq t \leq 15$ (blue part) and $15 \leq t \leq 35$ (red part), for $\alpha = 30$ and $\rho = 28$.

noted that the SD value is very similar. For the classic LSS, $SD = 1.00966$ and for DCLSS $SD = 0.99961$ which are both sufficiently close to 1.01.

### Comparison of LSS and DCLSS

In order to examine the effect of $\alpha$ in the precision of the SDs evaluated using DCLSS compared to classic LSS, the SD for different values of $\alpha$ was plotted in Fig. 2.27 for

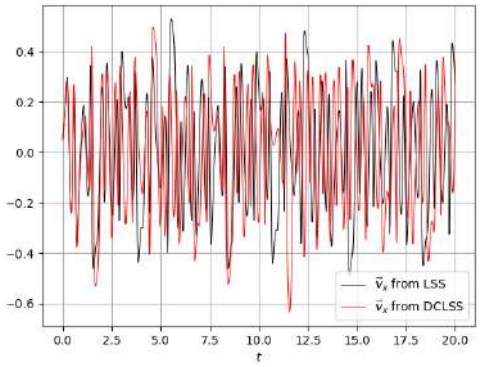



Figure 2.25: *Lorenz '63* Comparison of trajectory of $\vec{v}$ using LSS and DCLSS, for $a = 1$ and parameters found in the text.

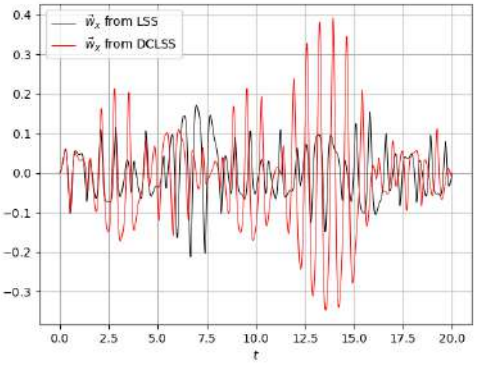



Figure 2.26: *Lorenz '63* Comparison of trajectory of $\vec{w}$ using LSS and DCLSS, for $a = 1$ and parameters found in the text.

LSS (in blue) and DCLSS (in red) for $T = 20$ time units, $\rho = 28$ and the parameters found in the text. Although the classic LSS is less sensitive to the choise of different ICs in this case, it is less accurate for larger values of $\alpha$ and begins to produce increasingly lower SD values than the reference of 1.01. The majority of the DCLSS runs however, for each $\alpha$ value are closer to the $SD = 1.01$ line and show no quality deterioration in the SD values produced for larger values of $\alpha$. Fig. 2.28 shows the SD values for $\rho$ values ranging from 0 to 100, using 20 random initial conditions at each one, for integration time $T = 20$ time units (in blue) and $T = 2000$ time units (in red).

### Consistency of the Solutions of the Boundary Value Problem using DCLSS

In order to examine the consistency of the solutions resulting from Eq. 2.28 compared to those from Eq. 2.29, the time-series $\vec{w}$ computed from Eq. 2.29 was used to evaluate time-series $\vec{v}$ from the discretized form of Eq. 2.28a, using the forward and backward Euler methods. The process is repeated twice, using as initial conditions for Eq. 2.28a the values $\vec{v}(0)$ and $\vec{v}(T)$ (for the forward and backward Euler method respectively).

Fig. 2.29 compares time-series $\vec{v}$ from the Euler method with that of DCLSS, using forward Euler. It can be observed that further away from the initial conditions, the trajectory diverges exponentially. However, as can be seen in Fig. 2.30, near the BC the two trajectories are very close to each other. Similar dynamics can be observed for the backward Euler method, in Figs. 2.31 and 2.32. Those findings suggest that regardless of the discretization scheme that is used, it is not feasible to solve Eq. 2.28a forward or backward in time without fixing its first and last values using BCs at $t = 0$ and $t = T$ (as is guaranteed by Eq. 2.29), because the chaotic nature

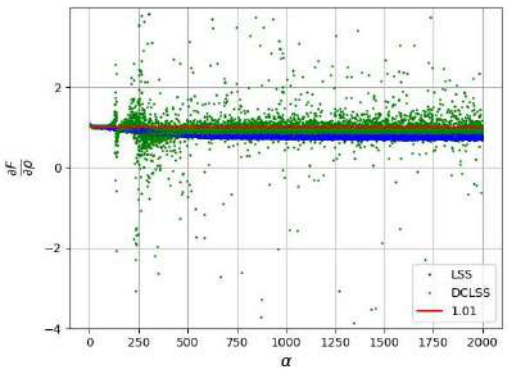



Figure 2.27: *Lorenz '63* The SD for various $\alpha$ values found using LSS (in blue) and DCLSS (in red) for $T = 20$ time units, $\rho = 28$ and the parameters found in the text.

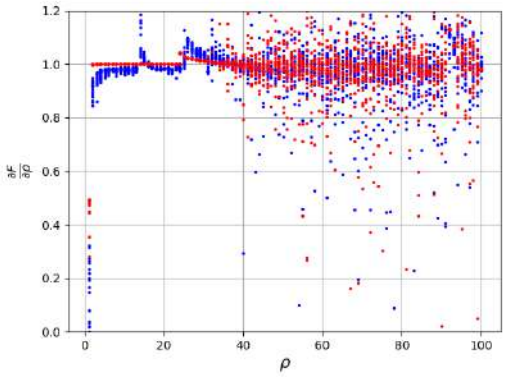



Figure 2.28: *Lorenz '63* The SD values for $\rho$ values ranging from 0 to 100, using 20 random initial conditions at each one, for integration time $T = 20$ time units (in blue) and $T = 2000$ time units (in red) and the parameters found in the text.

of the system renders it very sensitive to error accumulation during the forward or backward time marching.

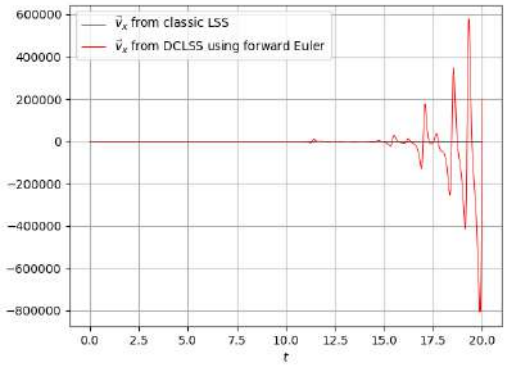



Figure 2.29: *Lorenz '63* Comparison of trajectory $\vec{v}$ using forward Euler method with time-series $\vec{w}$ known from DCLSS, for $a = 1$ and parameters found in the text.

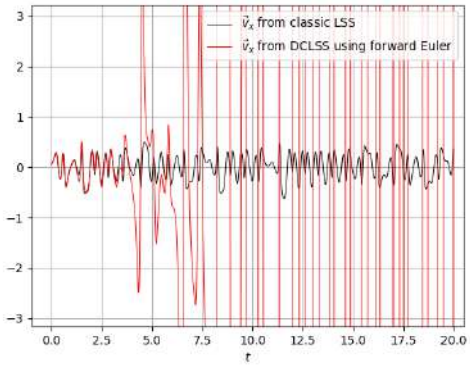



Figure 2.30: *Lorenz '63* Detail of the comparison of trajectory $\vec{v}$ using forward Euler method with time-series $\vec{w}$ known from DCLSS, for $a = 1$ and parameters found in the text.

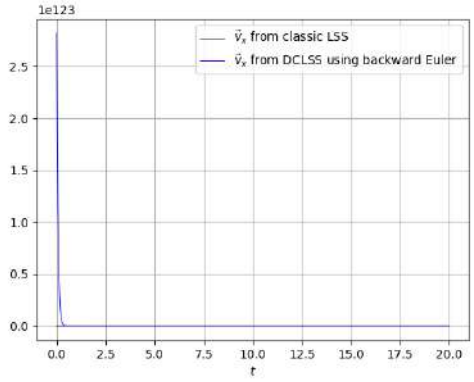



Figure 2.31: *Lorenz '63* Comparison of trajectory $\vec{v}$ using backward Euler method with time-series $\vec{w}$ known from DCLSS, for $a = 1$ and parameters found in the text.

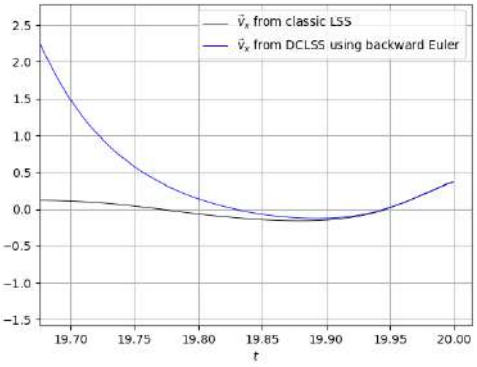



Figure 2.32: *Lorenz '63* Detail of the comparison of trajectory $\vec{v}$ using backward Euler method with time-series $\vec{w}$ known from DCLSS, for $a = 1$ and parameters found in the text.

# Chapter 3

# Application on the Van der Pol Problem

## 3.1 The Van der Pol Problem

The Van der Pol problem is another suitable example to demonstrate the failure of the Continuous Adjoint method to provide useful derivatives of long-time averaged quantities w.r.t. a design variable, and the superiority of the LSS algorithm to provide correct values, at lower cost compared to the Finite Differences method. It is chosen because of its challenging-to-capture dynamics, although it is not chaotic and it converges to a limit cycle, see Fig. 3.1. The problem is described by a single $2^{nd}$ order ODE that is given below:

$$\frac{d^2x}{dt^2} = -x + b(1 - x^2)\frac{dx}{dt} \tag{3.1}$$

Eq. 3.1 can be written as a system, and will be treated as one in this thesis:

$$\frac{dx}{dt} = y, \qquad x(0) = x_0 \tag{3.2a}$$

$$\frac{dy}{dt} = -x + b(1 - x^2)y, \qquad y(0) = y_0 \tag{3.2b}$$

The design variable is $b$, which lies within the interval $0.2 \leq b \leq 2.0$. Fig. 3.1 shows the solution to the Van der Pol Equation for $b = 1.2$, using a $4^{th}$ order Runge-Kutta method with $N = 50000$ time-steps and $\Delta t = 0.001$ time units. Fig. 3.2

illustrates the solutions $x(t)$ and $y(t)$ for $b = 2.0$, obtained using the same Runge-Kutta method.

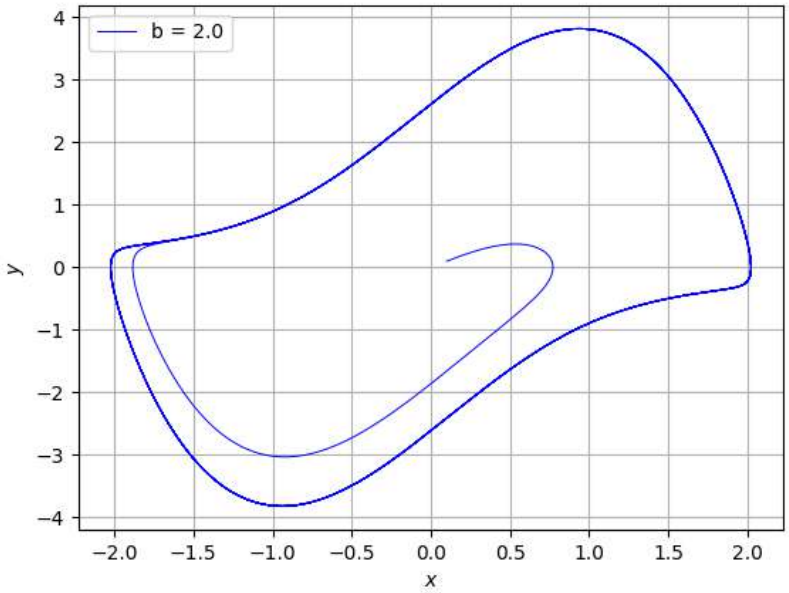



Figure 3.1: *Van der Pol* Solution for $b = 2.0$ and initial condition $\vec{u}_0 = [x_0, y_0]^T = [0.1, 0.1]^T$, using a $4^{th}$ order Runge-Kutta method with $N = 50000$ time-steps and $\Delta t = 0.001$ time units.

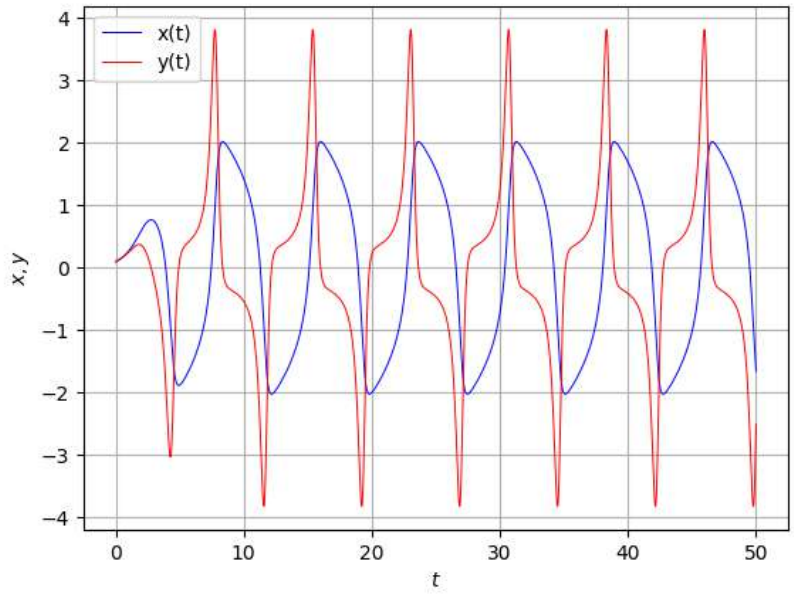



Figure 3.2: *Van der Pol* Solutions $x(t)$ (blue) and $y(t)$ (red) of the Van der Pol system for initial condition $\vec{u}_0 = [x_0, y_0]^T = [0.1, 0.1]^T$ for $b = 2.0$, using a $4^{th}$ order Runge-Kutta method with $N = 50000$ time-steps and $\Delta t = 0.001$ time units.

## 3.2 Continuous Adjoint Sensitivity Analysis

Throughout this analysis, for the purpose of demonstration, the objective function F is defined as:

$$F(b) = \left( \frac{1}{T} \int_0^T \left( y(t,b) \right)^8 \, dt \right)^{1/8} \tag{3.3}$$

F is the $L^P$ norm, where $p = 8$, of the long-time averaged $8^{th}$ power of $y$, and the design parameter vector is $\vec{b} = [b_0], b_0 = b$. In order to formulate the Continuous Adjoint method and compute the gradient of $F$ w.r.t. $\vec{b}$, the augmented objective function is defined as:

$$F_{aug} = F + \int_0^T \vec{\Psi}^T \vec{R} \, dt \tag{3.4}$$

where $\vec{\Psi} = [\Psi_x, \Psi_y]^T \in \mathbb{R}^2$ is the adjoint variable vector field and $\vec{R} \in \mathbb{R}^2$ are the residuals of the now two Van der Pol equations:

$$\vec{R} = \begin{pmatrix} \frac{dx}{dt} - y \\ \frac{dy}{dt} + x - b(1 - x^2)y \end{pmatrix} \tag{3.5}$$

Eq. 2.3 is then written as:

$$F_{aug} = \left(\frac{1}{T}\int_0^T \left(y(t,b)\right)^8 \, dt\right)^{1/8} + \int_0^T \Psi_x \left(\frac{dx}{dt} - y\right) dt + \int_0^T \Psi_y \left(\frac{dy}{dt} + x - b(1-x^2)y\right) dt \tag{3.6}$$

Differentiating Eq. 3.6 w.r.t. $b$ and since the operators $\frac{\delta}{\delta b}$ and $\frac{d}{dt}$ are interchangeable, results in:

$$\frac{\delta F_{aug}}{\delta b} = \frac{\delta F}{\delta b} + \int_0^T \Psi_x \frac{d}{dt}\left(\frac{\delta x}{\delta b}\right) dt + \int_0^T \Psi_y \frac{d}{dt}\left(\frac{\delta y}{\delta b}\right) dt + \int_0^T \Psi_x \left(-y\right) dt + \int_0^T \Psi_y \left(x - b(1-x^2)y\right) dt \tag{3.7}$$

By integrating the time derivative terms by parts, for $u_k = x, y$ and $\Psi_k = \Psi_x, \Psi_y$ respectively:

$$\int_0^T \Psi_k \frac{d}{dt}\left(\frac{\delta u_k}{\delta b}\right) \, dt = \left[\Psi_k \frac{\delta u_k}{\delta b}\right]_0^T - \int_0^T \frac{d\Psi_k}{dt}\frac{\delta u_k}{\delta b} \, dt \tag{3.8}$$

It is also true that:

$$\frac{\delta F}{\delta b} = \frac{1}{8}\left(\frac{1}{T}\int_0^T y^8 \, dt\right)^{-7/8} \left(\int_0^T 8y^7 \frac{\delta y}{\delta b} \, dt\right) \tag{3.9}$$

After factoring out terms $\frac{\delta u_k}{\delta b}$, Eq. 3.7 results in:

$$\begin{aligned}
\frac{\delta F_{aug}}{\delta b} = & \int_0^T \frac{\delta x}{\delta b}\left(-\frac{d\Psi_x}{dt} + \Psi_y\left(1 + 2bxy\right)\right) \, dt \\
& + \int_0^T \frac{\delta y}{\delta b}\left(-\frac{d\Psi_y}{dt} - b(1-x^2)\Psi_y - \Psi_x + \frac{1}{T}y^7\left(\frac{1}{T}\int_0^T y^8 \, dt\right)^{-7/8}\right) \, dt \\
& + \left[\Psi_x \frac{\delta x}{\delta b}\right]_0^T + \left[\Psi_y \frac{\delta y}{\delta b}\right]_0^T \\
& - \int_0^T y\Psi_y\left(1 - x^2\right) \, dt
\end{aligned} \tag{3.10}$$

The Field Adjoint Equations (FAEs) are determined by setting the multipliers of $\frac{\delta u_k}{\delta b}$ to zero continuously throughout the interval $0 \leq t < T$. The Adjoint Boundary Conditions (ABCs) that accompany the FAEs are determined by setting the

coefficients of terms $\frac{\delta u_k}{\delta b}$ to zero at $t = T$. The FAEs and ABCs are:

$$\frac{d\Psi_x}{dt} = \Psi_y \left(1 + 2bxy\right) \tag{3.11a}$$

$$\frac{d\Psi_y}{dt} = -b(1-x^2)\Psi_y - \Psi_x + \frac{1}{T}y^7 \left(\frac{1}{T}\int_0^T y^8 \, dt\right)^{-7/8} \tag{3.11b}$$

$$\Psi_x(T) = \Psi_y(T) = 0 \tag{3.11c}$$

which imply that, as is always the case in unsteady adjoint, the adjoint equations should be integrated backward in time. After eliminating the terms $\frac{\delta u_k}{\delta b}$ by satisfying the FAEs and ABCs, the only remaining term on the right-hand-side of Eq. 3.10 constitutes the Sensitivity Derivative (SD):

$$\frac{\delta F}{\delta b} = -\int_0^T y\Psi_y \left(1 - x^2\right) \, dt \tag{3.12}$$

To solve the FAEs the same $4^{th}$ order Runge-Kutta method is used to integrate Eqs. 3.11 backward in time with, of course, the same time step $\Delta t = 0.001$. After having computed the adjoint fields, the SD results from the integral of Eq. 3.12. The objective for $0.2 \leq b \leq 2.0$, $T = 50$ and the parametric values mentioned in the text, is seen in Fig. 3.3. Fig. 3.4 shows $F$ for different values of integration time $T$ and for several values of $b$. The value of the objective converges at $T = 1000$ t.u. and remains essentially the same for larger $T$. The SD for $0.2 \leq b \leq 2.0$ and $T = 50$ is seen in Fig. 3.5. In both figures, for each value of $b$, 20 random initial conditions in the interval $0 < x_0, y_0 < 1$ are plotted. The SD values found using the CA method are meaningless, and follow a pattern of exponential increase similar to the Lorenz '63 case. The findings are consistent with those of [34].

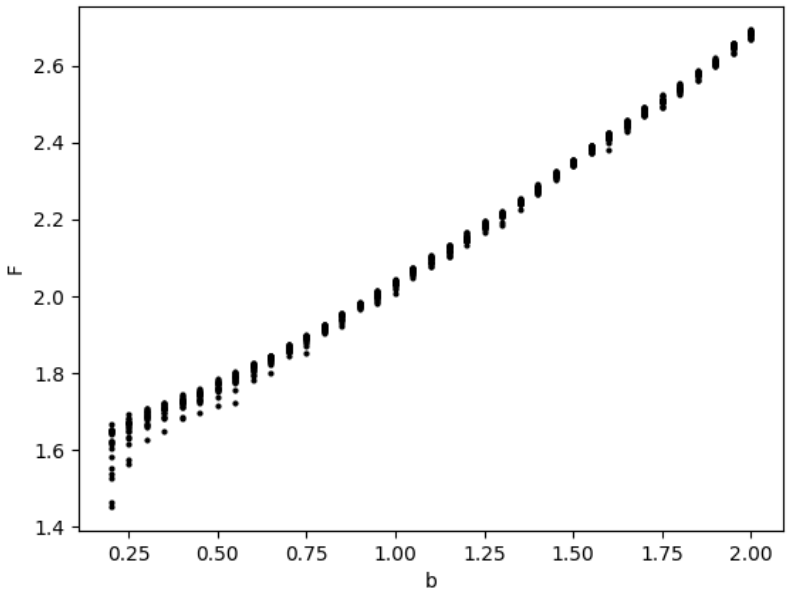



Figure 3.3: *Van der Pol* The objective function value for $0.2 \leq b \leq 2.0$, $T = 50$ and the parametric values mentioned in the text. For each $b$, 20 random initial conditions are plotted

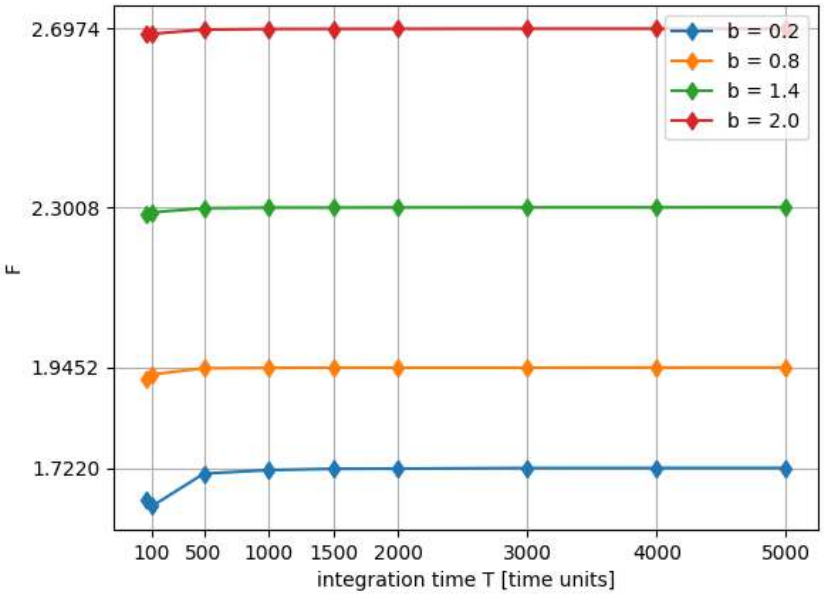



Figure 3.4: *Van der Pol* The objective function value for $0.2 \leq b \leq 2.0$, $T = 50$ and the parametric values mentioned in the text. For each $b$, 20 random initial conditions are plotted

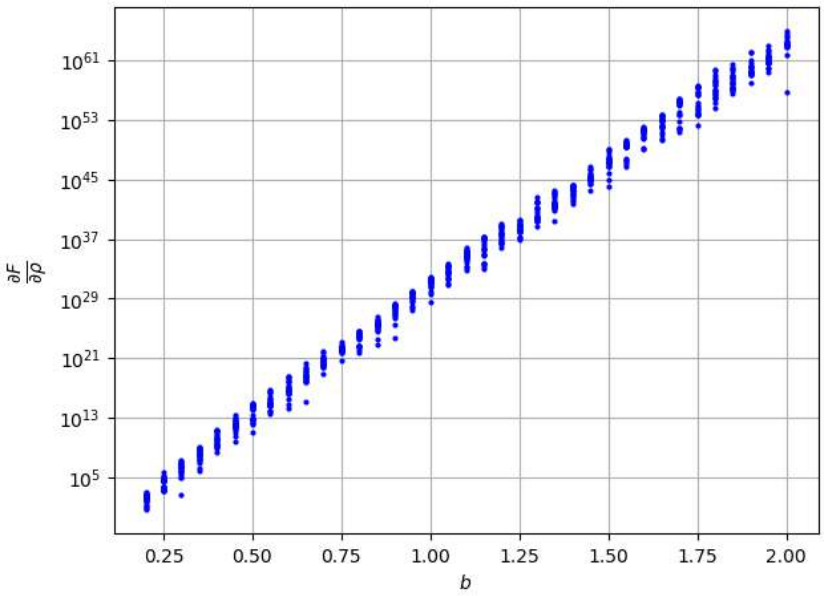



Figure 3.5: *Van der Pol* The SD value for $0.2 \leq b \leq 2.0$, $T = 50$ and the parametric values mentioned in the text, using the CA method. For each $b$, 20 random initial conditions are plotted

## 3.3 Finite Differences Sensitivity Analysis

The second method used to compute the sensitivity derivative of Eq. 3.3 w.r.t. $b$ is the Finite Differences (FD) method. The same range was used for $b$, from 0.2 to 2.0, with perturbation $\Delta b = 0.05$. The formula for the second-order, central FD method is seen in Eq. 2.12.

In Fig. 3.7 the SD is plotted for $0.2 \leq b \leq 2.0$ and for several values of $\delta b$. The SD values for $\delta b < 0.05$ are meaningless, which is consistent to the trend observed in the Lorenz '63 case, that in order for the SDs to be accurate, $\delta b$ has to be very large compared to what is customarily used. This appears to be the case, because as the denominator of the right-hand-side of Eq. 2.12 decreases, the numerator does not also decrease, as it should for the SD value to be correct. This implies that infinitesimal changes in $b$ do not correspond to infinitesimal changes in the SD, but rather significant ones, which is attributed to the unpredictable nature (not quite chaotic though) of the Van der Pol system. In Fig. 3.6 the SD for $0.2 \leq b \leq 2.0$ is plotted for integration time of $T = 50$ (in blue), $T = 5000$ (in red) time units and $\delta b = 0.05$, using 20 random initial conditions for each value of $b$. It is obvious that in order to get accurate SDs, the integration time $T$ has to be impractically large.

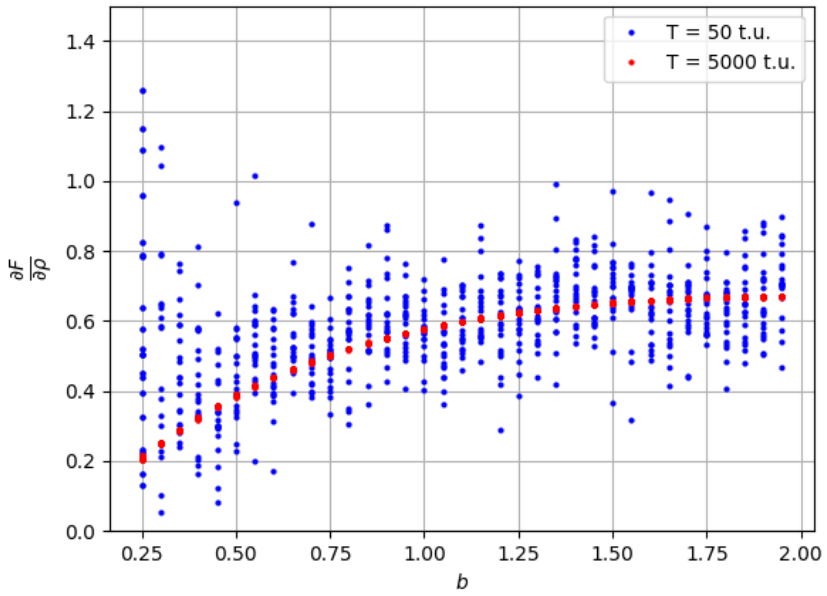



Figure 3.6: *Van der Pol* The SD value for $0.2 \leq b \leq 2.0$, where $T = 50$ (blue) and $T = 5000$ (red), and the parametric values mentioned in the text, using the FD method. For each $b$, 20 random initial conditions are plotted.

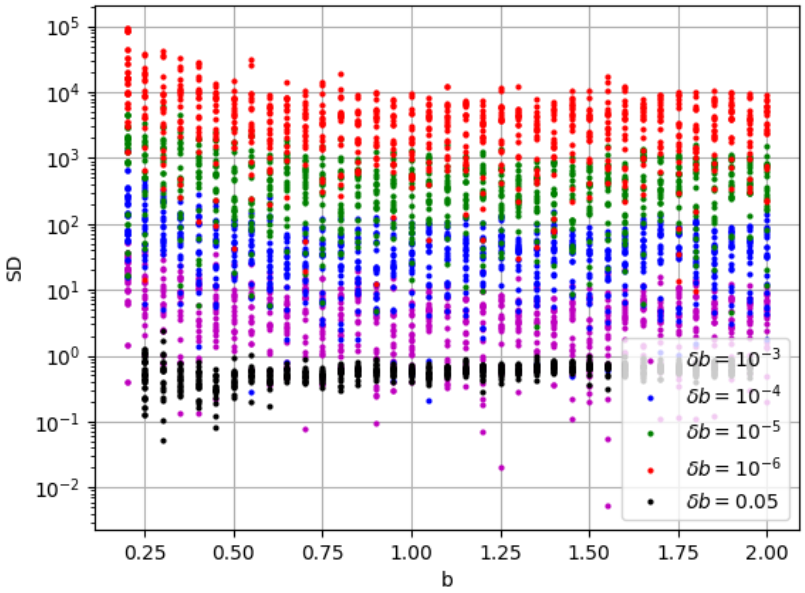



Figure 3.7: *Van der Pol* The SD value for $0.2 \leq b \leq 2.0$, where $T = 50$, for different values of $\delta b$, and the parametric values mentioned in the text, using the FD method. For each $b$, 20 random initial conditions are plotted.

## 3.4 Sensitivity Analysis with Classic vs Discretely Consistent LSS

In this section, the original version of LSS will be compared to DCLSS in order to evaluate the SD of Eq. 3.3 w.r.t. $b$. The analysis and development of equations is already presented in a generalized manner for the case of the Lorenz '63 problem. In the Van der Pol problem, the coefficients of Eqs. A.18 and A.50 are derived in Appendix B. In Fig. 3.8 the SD for $0.2 \leq b \leq 2.0$ is evaluated for integration time $T = 50$ time units using LSS (in blue) and DCLSS (in red). It is apparent that the Discretely Consistent version of the LSS algorithm (DCLSS) is much more precise than classic LSS, and it produces SD values with minimal deviation for different ICs, which is a very important attribute of the algorithm in terms of avoiding extra computational cost while achieving very high precision.

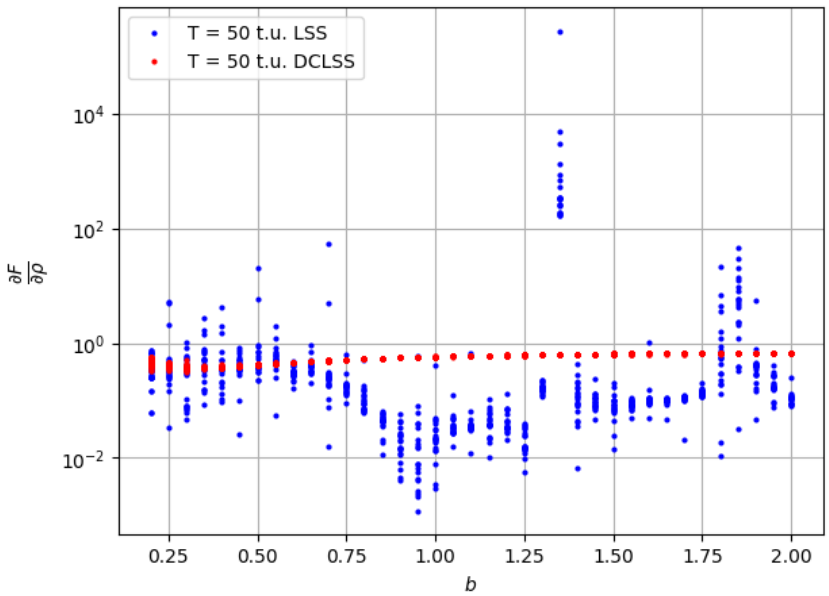



Figure 3.8: *Van der Pol* The SD value for $0.2 \leq b \leq 2.0$, where $T = 50$ time units, and the parametric values mentioned in the text, using LSS (in blue) and DCLSS (in red). For each $b$, 20 random initial conditions are plotted.

### 3.4.1 Consistency of the Solutions of the Boundary Value Problem using DCLSS

A similar analysis to 2.4.6, regarding the consistency of the Solutions of the Boundary Value Problem using DCLSS, is performed for the Van der Pol system. Fig. 3.9 shows the first component $v_x$ of the time-series vector $\vec{v}$ using DCLSS (in black dots), and using the forward (in blue) and backward (in red) Euler method. The time-series are identical with one another, which translates to the $2^{nd}$ order Eq. 2.29 being consistent with the system of Eq. 2.28. The previous findings indicate that, contrary to the Lorenz '63 and Rossler systems, the non-chaotic nature of the Van der Pol equations allows the system of Eq. 2.28 to be solved forward and backward in time without the need for the double bounding enforced by Eq. 2.28.

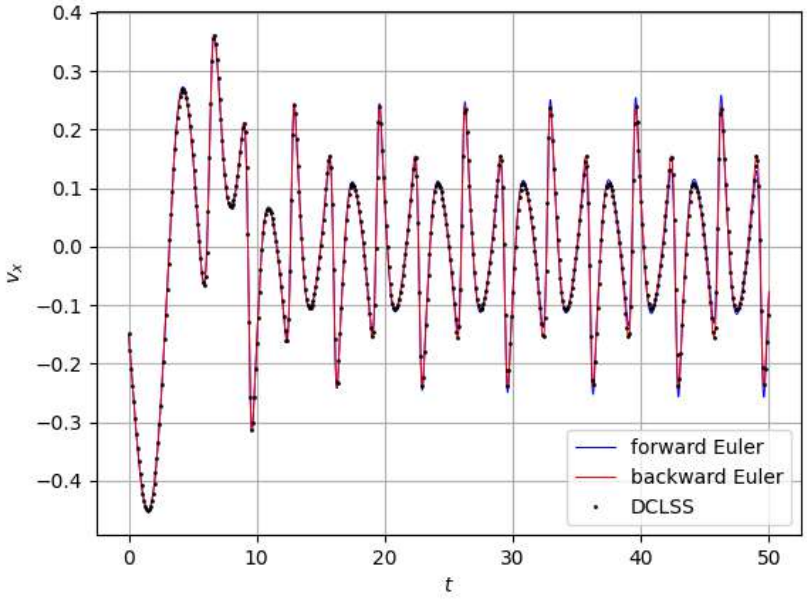



Figure 3.9: *Van der Pol* Comparison of $v_x$ evaluated using DCLSS (in black dots), forward (in blue) and backward (in red) Euler.

# Chapter 4

# Application on the Rössler Problem

## 4.1 The Rössler Problem

The Rössler problem is yet another suitable example to demonstrate the failure of the Continuous Adjoint method to provide useful derivatives of long-time averaged quantities w.r.t. a design variable, and the superiority of the LSS algorithm to provide correct values, at lower cost compared to the Finite Differences method. The problem is described by a system of three ODEs that is given below:

$$\frac{dx}{dt} = -y - z, \qquad x(0) = x_0 \tag{4.1a}$$

$$\frac{dy}{dt} = x + ay, \qquad y(0) = y_0 \tag{4.1b}$$

$$\frac{dz}{dt} = b + z(x - c), \qquad z(0) = z_0 \tag{4.1c}$$

The design variable is $a$, which lies within the interval $0.1 \leq a \leq 0.4$. Fig. 4.1 shows the solution to the Rössler Equation for $a = 0.2$, using a $4^{th}$ order Runge-Kutta method with $N = 1000000$ time-steps and $\Delta t = 0.001$ time units. The integration time $T = Ndt = 1000$ time units is quite large compared to the previous cases. That is because the Rössler system has larger time-scales (pseudo-period) than the Lorenz '63 or the Van der Pol systems, and in order to capture the system's dynamics, the integration time has to be significantly larger than its largest time-scale. Fig. 4.2 illustrates the chaotic behavior of the system by comparing $z(t)$ for two very similar

design variable values $a = 0.20$ and $a = 0.21$, obtained using the same Runge-Kutta method. Initially, the two solution trajectories are very close to each other. After that, however, due to the chaotic nature of the system, they differ greatly.

The behavior of the Rössler system for different $a$ values is the following:

1. $a \in (-\infty, 0]$ : Stable fixed point attractor (NOT chaotic).
2. $a = 0.1$ : Unit cycle of period 1 time unit (NOT chaotic).
3. $a \in (0.13, \infty)$ : Chaotic, increasingly so for larger $a$ values.

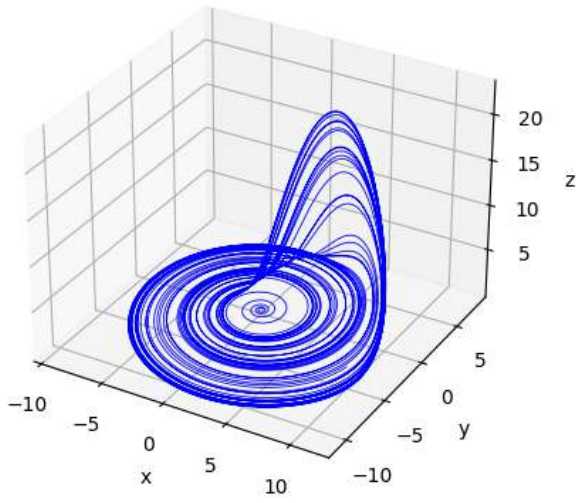



Figure 4.1: *Rössler* Solution for $a = 0.2$ and initial condition $\vec{u}_0 = [x_0 y_0 z_0]^T = [0.10.10.1]^T$, using a $4^{th}$ order Runge-Kutta method with $N = 300000$ time-steps and $\Delta t = 0.001$ time units.

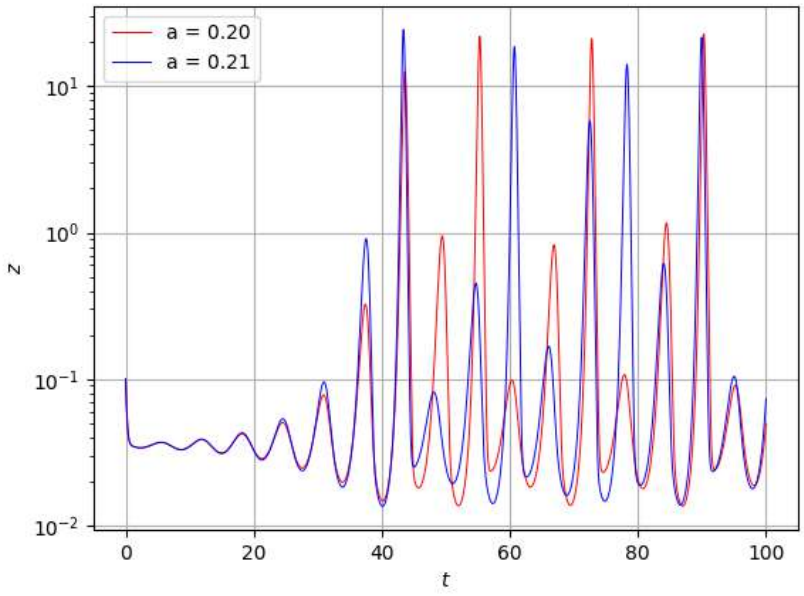



Figure 4.2: *Rössler* Solution for initial condition $\vec{u}_0 = [x_0 y_0 z_0]^T = [0.10.10.1]^T$ and for $a = 0.20$ (red) and $a = 0.21$ (blue), using a $4^{th}$ order Runge-Kutta method with $N = 100000$ time-steps and $\Delta t = 0.001$ time units. The difference is subtle at first, but it increases with $t$, as is expected from the chaotic nature of the system.

## 4.2 Continuous Adjoint Sensitivity Analysis

Throughout this analysis, for the purpose of demonstration, the objective function F is defined as:

$$F(b) = \frac{1}{T}\int_0^T z(t,b)\,dt \tag{4.2}$$

F is the long-time averaged $z$, and the design parameter vector is $\vec{b} = [b_0]\,, b_0 = a$. In order to formulate the Continuous Adjoint method and compute the gradient of $F$ w.r.t. $\vec{b}$, the augmented objective function is defined as:

$$F_{aug} = F + \int_0^T \vec{\Psi}^T \vec{R}\,dt \tag{4.3}$$

where $\vec{\Psi} = [\Psi_x \Psi_y \Psi_z]^T \in \mathbb{R}^3$ is the adjoint variable vector field and $\vec{R} \in \mathbb{R}^3$ are the residuals of the three Rössler equations:

$$\vec{R} = \begin{pmatrix} \frac{dx}{dt} + y + z \\ \frac{dy}{dt} - x - ay \\ \frac{dy}{dt} - b - z(x - c) \end{pmatrix} \tag{4.4}$$

Eq. 4.3 is then written as:

$$\begin{aligned} F_{aug} = \frac{1}{T} \int_0^T z(t, b)\, dt + \int_0^T \Psi_x \left( \frac{dx}{dt} + y + z \right) dt \\ + \int_0^T \Psi_y \left( \frac{dy}{dt} - x - ay \right) dt + \int_0^T \Psi_z \left( \frac{dz}{dt} - b - z(x - c) \right) dt \end{aligned} \tag{4.5}$$

Differentiating Eq. 4.5 w.r.t. $a$, and since the operators $\frac{\delta}{\delta a}$ and $\frac{d}{dt}$ are interchangeable, results in:

$$\begin{aligned} \frac{\delta F_{aug}}{\delta b} = \frac{\delta F}{\delta a} + \int_0^T \Psi_x \frac{d}{dt} \left( \frac{\delta x}{\delta a} \right) dt + \int_0^T \Psi_y \frac{d}{dt} \left( \frac{\delta y}{\delta a} \right) dt + \int_0^T \Psi_z \frac{d}{dt} \left( \frac{\delta z}{\delta a} \right) dt \\ + \int_0^T \Psi_x \left( y + z \right) dt + \int_0^T \Psi_y \left( -x - ay \right) dt + \int_0^T \Psi_z \left( -b - z(x - c) \right) dt \end{aligned} \tag{4.6}$$

Integrating by parts, for $u_k = x, y, z$ and $\Psi_k = \Psi_x, \Psi_y, \Psi_z$, yields:

$$\int_0^T \Psi_k \frac{d}{dt} \left( \frac{\delta u_k}{\delta a} \right) dt = \left[ \Psi_k \frac{\delta u_k}{\delta a} \right]_0^T - \int_0^T \frac{d\Psi_k}{dt} \frac{\delta u_k}{\delta a}\, dt \tag{4.7}$$

After factoring out terms $\frac{\delta u_k}{\delta a}$, Eq. 4.6 results in:

$$\begin{aligned} \frac{\delta F_{aug}}{\delta \rho} = \int_0^T \frac{\delta x}{\delta a} \left( -\frac{d\Psi_x}{dt} - \Psi_y - \Psi_z z \right) dt + \int_0^T \frac{\delta y}{\delta a} \left( -\frac{d\Psi_y}{dt} + \Psi_x - a\Psi_y \right) dt \\ + \int_0^T \frac{\delta z}{\delta a} \left( -\frac{d\Psi_z}{dt} \Psi_x - \Psi_z (x - c) + \frac{1}{T} \right) dt + \left[ \Psi_x \frac{\delta x}{\delta a} \right]_0^T + \left[ \Psi_y \frac{\delta y}{\delta a} \right]_0^T + \left[ \Psi_z \frac{\delta z}{\delta a} \right]_0^T \\ - \int_0^T \Psi_y y\, dt \end{aligned} \tag{4.8}$$

The Field Adjoint Equations (FAEs) are determined by setting the multipliers of $\frac{\delta u_k}{\delta a}$ to zero continuously throughout the interval $0 \leq t < T$. The Adjoint Boundary Conditions (ABCs) that accompany the FAEs are determined by setting the

coefficients of terms $\frac{\delta u_k}{\delta a}$ to zero at $t = T$. The FAEs and ABCs are:

$$\frac{d\Psi_x}{dt} = -\Psi_y - \Psi_z z \tag{4.9a}$$

$$\frac{d\Psi_y}{dt} = +\Psi_x - a\Psi_y \tag{4.9b}$$

$$\frac{d\Psi_y}{dt} = \Psi_x - \Psi_z(x - c) + \frac{1}{T} \tag{4.9c}$$

$$\Psi_x(T) = \Psi_y(T) = \Psi_z(T) = 0 \tag{4.9d}$$

which imply that, as is always the case in unsteady adjoint, the adjoint equations should be integrated backward in time. After eliminating the terms $\frac{\delta u_k}{\delta a}$ by satisfying the FAEs and ABCs, the only remaining term on the right-hand-side of Eq. 4.8 constitutes the Sensitivity Derivative (SD):

$$\frac{\delta F}{\delta a} = -\int_0^T \Psi_y y \, dt \tag{4.10}$$

To solve the FAEs the same $4^{th}$ order Runge-Kutta method is used to integrate Eqs. 4.8 backward in time with, of course, the same time step $\Delta t = 0.001$. After having computed the adjoint fields, the SD results from the integral of Eq. 4.10. The objective for $0.1 \leq b \leq 0.4$, $T = 1000$ (in blue) and $T = 10000$ (in red) and the parametric values mentioned in the text, is seen in Fig. 4.3. The SD for those values of $a$ and $T = 1000$ time units is seen in Fig. 4.4. In both figures, for each value of $a$, 20 random initial conditions in the interval $0 < x_0, y_0, z_0 < 1$ are plotted. The SD values from the CA method for $0.1 \leq a \leq 0.13$, where the Rössler system is not chaotic, are meaningful although false, as will be discussed in section 4.5, where for comparison, they are plotted against those from FD and LSS. However, for $a > 0.13$, where the system begins to present rapidly increasing chaotic behavior, the SD values computed using the CA method are increasingly meaningless, diverging to infinity, which is a behavior consistent with the previous cases.

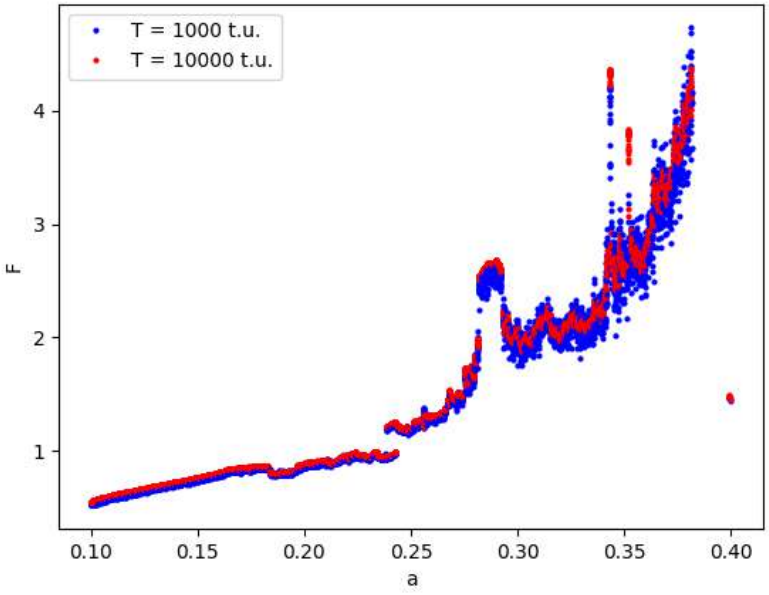



Figure 4.3: *Rössler* F for $0.1 \leq b \leq 0.4$, $T = 1000$ (in blue) and $T = 10000$ (in red) and the parametric values mentioned in the text.

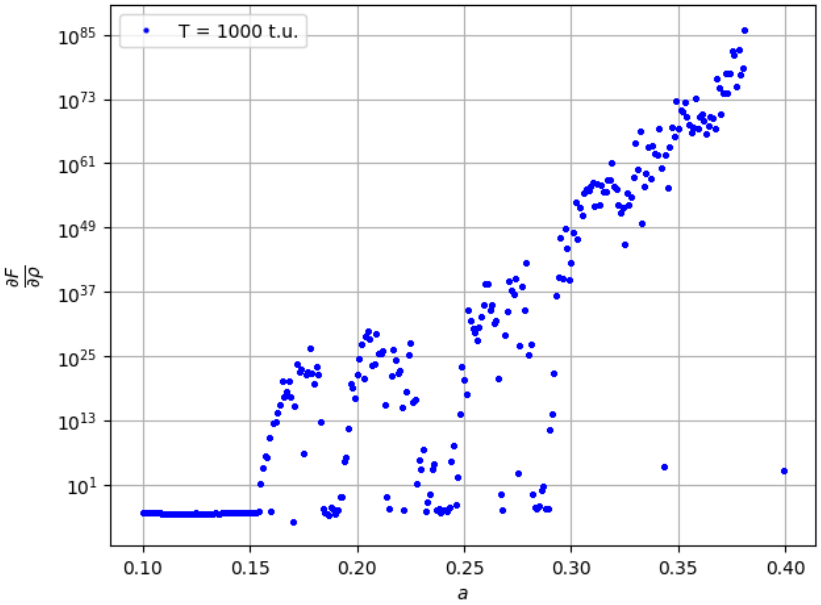



Figure 4.4: *Rössler* The SD for $0.1 \leq a \leq 0.4$ using CA, for the parametric values mentioned in the text

## 4.3 Finite Differences Sensitivity Analysis

The second method used to compute the sensitivity derivative of Eq. 4.2 w.r.t. $a$ is the Finite Differences (FD) method. The same range was used for $a$, from 0.1 to 0.4, with perturbation $\Delta a = 0.001$. The formula for the second-order, central FD method is seen in Eq. 2.12.

In Fig. 4.5 the SD for $0.1 \leq a \leq 0.4$ is plotted for integration time of $T = 1000$ (in blue) and $T = 10000$ (in red) time units, using 20 random initial conditions for each value of $a$. The values produced are meaningful throughout the interval of $0.1 \leq a \leq 0.4$, but it is obvious that in order to get accurate SDs, the integration time $T$ has to be impractically large. For larger values of $a$, there is increased variation of the SD value, with extreme "hills" and "valleys". This is attributed to the rapidly increasing chaotic nature of the system and not to a possible failure of the FD method, as even with an extremely large integration time ($T = 1000$ time units), there is no mitigation of the phenomenon.

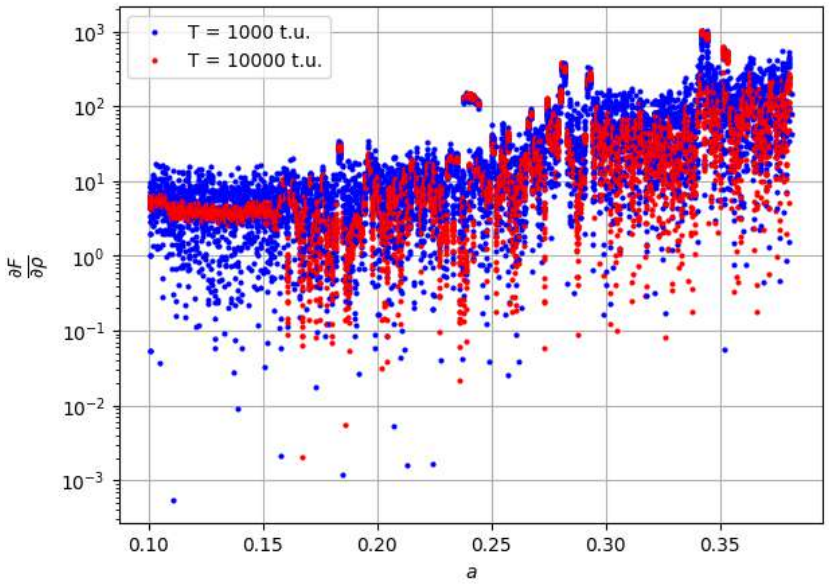



Figure 4.5: *Rössler* The SD value for $0.1 \leq a \leq 0.4$, where $T = 1000$ (blue) and $T = 10000$ (red), and the parametric values mentioned in the text, using the FD method. For each $a$, 20 random initial conditions are plotted.

## 4.4 Sensitivity Analysis with Classic vs Discretely Consistent LSS

In this section, the original version of LSS will be compared to DCLSS in order to evaluate the SD of Eq. 4.2 w.r.t. $a$. The analysis and development of equations is already presented in a generalized manner for the case of the Lorenz '63 problem. In the Rössler problem, the coefficients of Eqs. A.18 and A.50 are derived in Appendix C. In Fig. 4.6 the SD for $0.1 \leq a \leq 0.4$ is evaluated for integration time $T = 1000$ time units, using LSS (in blue) and DCLSS (in red). For $0.1 \leq a \leq 0.13$, where the system is not chaotic, the LSS and DCLSS methods produce meaningless SD values. However, for larger $a$ values where the Rössler system is chaotic, they produce meaningful ones. DCLSS has less variation in the SD values due to different ICs, which means it is more precise than classic LSS. A better comparison between the two variations of the algorithm is made with a detailed view in Fig. 4.7.

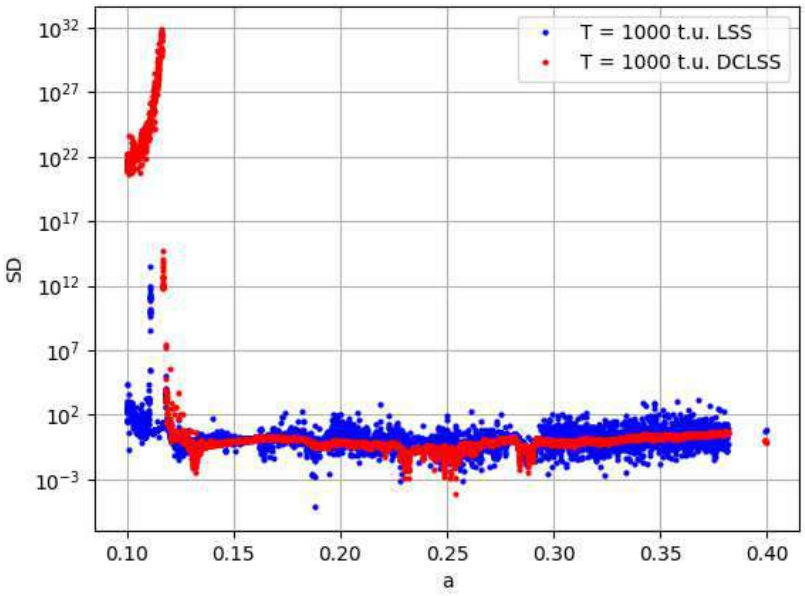



Figure 4.6: *Rössler* The SD value for $0.1 \leq a \leq 0.4$, where $T = 1000$ time units, and the parametric values mentioned in the text, using LSS (in blue) and DCLSS (in red). For each $a$, 20 random initial conditions are plotted.

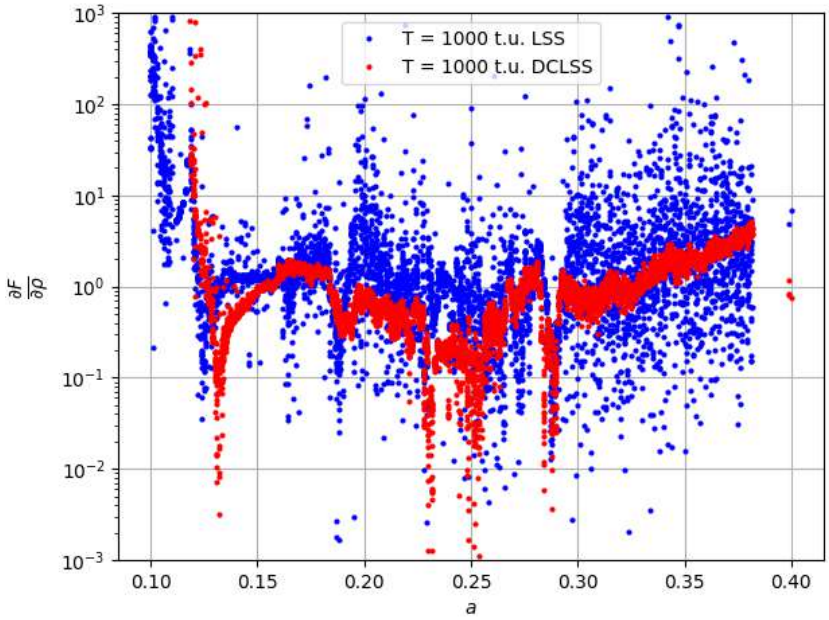



Figure 4.7: *Rössler* The SD value for $0.1 \leq a \leq 0.4$, where $T = 1000$ time units, and the parametric values mentioned in the text, using LSS (in blue) and DCLSS (in red). For each $a$, 20 random initial conditions are plotted.

### 4.4.1 Consistency of the Solutions of the Boundary Value Problem using DCLSS

A similar analysis to 2.4.6, regarding the consistency of the Solutions of the Boundary Value Problem using DCLSS, is performed for the Rössler system as well. Fig. 4.8 shows the first component $v_x$ of the time-series vector $\vec{v}$ using DCLSS (in black) and forward Euler (in blue). Also, Fig. 4.9 shows the first component $v_x$ of the time-series vector $\vec{v}$ using DCLSS (in black) and backward Euler (in blue). In both figures, it is evident that, due to the extremely chaotic nature of the Rössler system, it is not feasible to solve Eq. 2.28a forward or backward in time without fixing its first and last values using BCs at $t = 0$ and $t = T$, as is guaranteed by Eq. 2.29. These findings are consistent with those of the Lorenz '63 case.

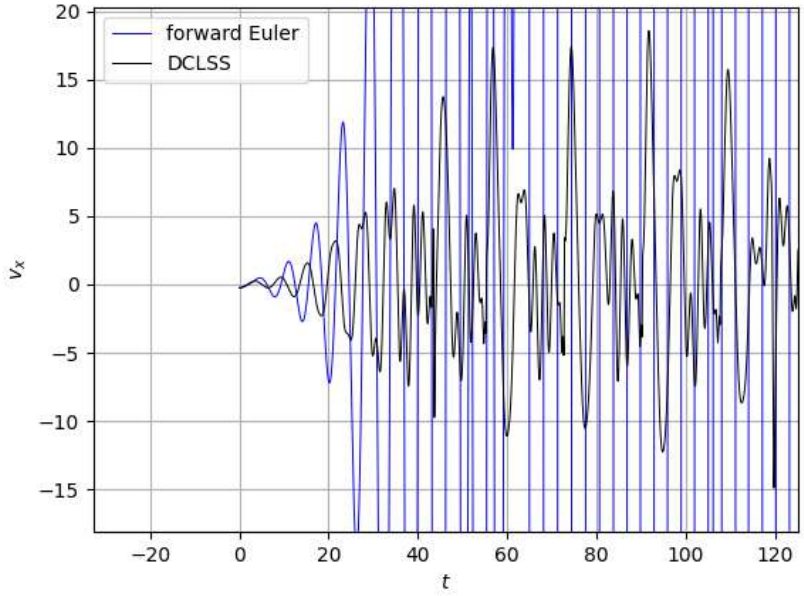



Figure 4.8: *Rössler* Comparison of $v_x$ evaluated using DCLSS (in black) and forward Euler (in blue).

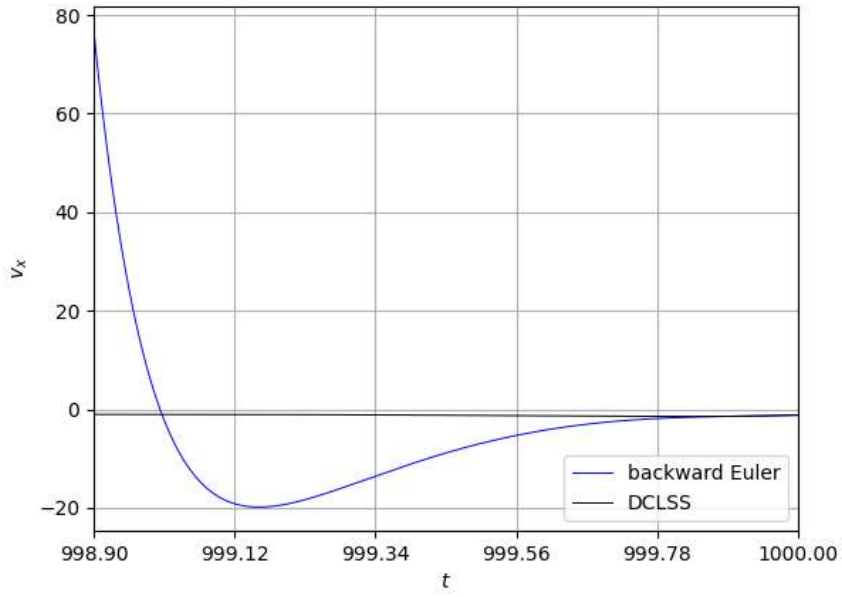



Figure 4.9: *Rössler* Comparison of $v_x$ evaluated using DCLSS (in black) and backward Euler (in blue).

## 4.5 Conclusions

A very important observation that was made is that in the interval $0.1 \leq a \leq 0.13$ where the Rössler system is not chaotic, the CA and FD methods produced meaningful SD values, and in $0.13 \leq a \leq 0.4$ where the Rössler system is chaotic, the CA failed to produce meaningful SD values. However, only the SD values from FD seem to be correct. On the contrary, the classic and Discretely Consistent LSS methods presented the exact opposite behavior, meaning that they failed produce meaningful SD values for $0.1 \leq a \leq 0.13$, but succeeded for $0.13 \leq a \leq 0.4$, and DCLSS even more precisely so. In Fig. 4.10 the SD value was plotted for $0.1 \leq a \leq 0.13$ using the CA, FD and LSS methods, for $T = 1000$ time units and parameter values found in the text. Only the SDs produced using the LSS method seem to be in accordance with those from FD. Fig. 4.11 shows the SD value for $0.13 \leq a \leq 0.4$ where the Rössler system is chaotic, using the FD, LSS and DCLSS methods, for $T = 1000$ time units and parameter values found in the text. The DCLSS method is more accurate and presents the least variation due to random initial conditions. It should be emphasized that, as can be seen in Figs. 4.10 and 4.11, due to the extreme chaotic nature of the Rössler system, the SD values produced by all methods are very sensitive to parameter changes.

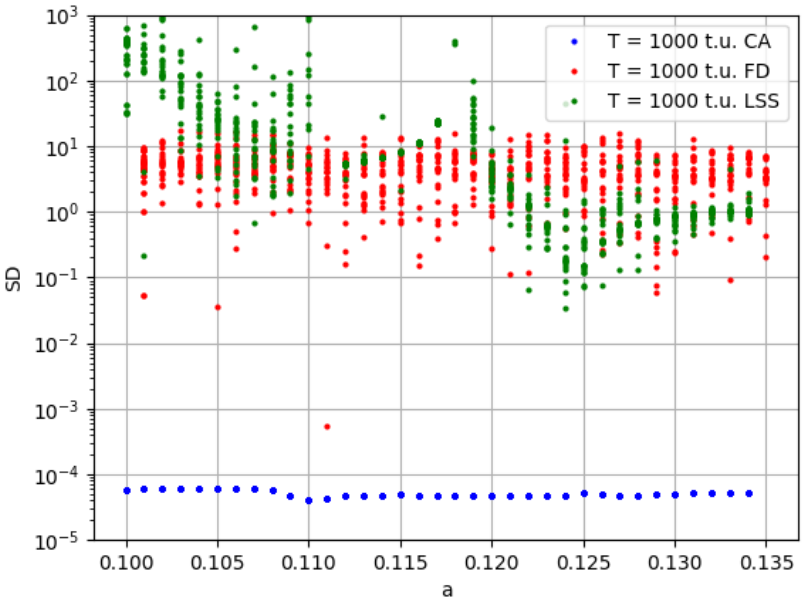



Figure 4.10: *Rössler* The SD value for $0.1 \leq a \leq 0.13$, where $T = 1000$ time units, and the parametric values mentioned in the text, using CA (in blue), FD (in red) and LSS (green). For each $a$, 20 random initial conditions are plotted.

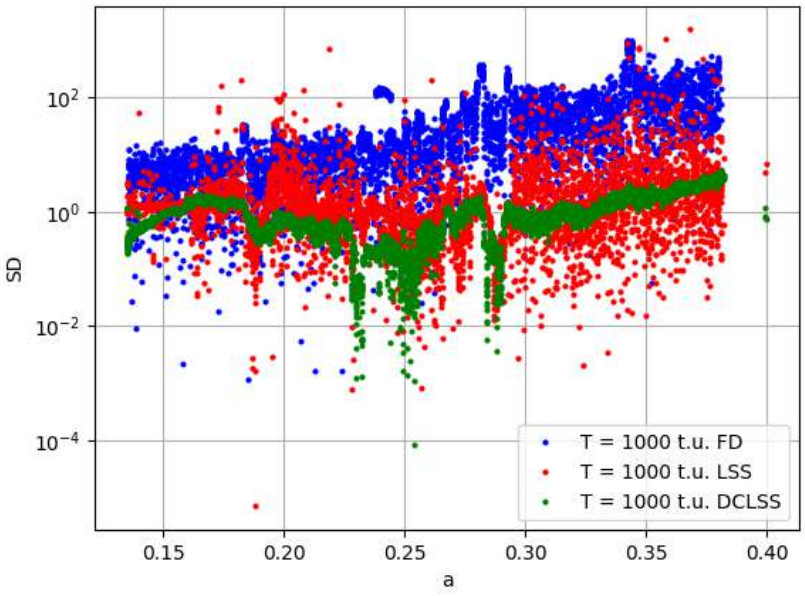



Figure 4.11: *Rössler* The SD value for $0.13 \leq a \leq 0.4$, where $T = 1000$ time units, and the parametric values mentioned in the text, using FD (in blue), LSS (in red) and DCLSS (green). For each $a$, 20 random initial conditions are plotted.

algorithms

# Chapter 5

# Data Assimilation

## 5.1 Increased Prediction Accuracy with Data Assimilation

Data Assimilation [27, 12, 20, 31, 11] is a technique that combines both observational (experimental) data and mathematical (theoretical) models to improve the accuracy and reliability of predictions about a system's state. In real-world applications, experimental measurements are often noisy and prone to errors, while simulations based on mathematical models can be sensitive to assumptions and approximations that may not fully capture the complexities of the system. Because of these limitations, relying solely on one source—either the measurements or the model—can lead to suboptimal results.

Data assimilation works by integrating observational data into the model in a way that allows both data sources to inform and correct each other. This iterative process improves the model's predictive power and enhances the analysis of the system's current and future states. Essentially, data assimilation provides a framework where observations constrain and correct model predictions, while the model offers structure and continuity to the often sparse or irregularly spaced data measurements.

This approach is particularly powerful in fields like meteorology, oceanography, and environmental science, where it's common to have an underlying set of governing equations (such as the Navier-Stokes equations in CFD) along with large sets of observational data. In these fields, data assimilation can be used to refine state estimates of the atmosphere, ocean, or other complex systems by continuously updating model states with new observations as they become available. In general, data assimilation is applied to all kinds of physical systems that are imperfectly

modeled, both chaotic and non-chaotic, where measurements have the potential to compensate for that loss of information and contribute to the evaluation of a more precise estimate of the true state.

One of the key concepts behind data assimilation is that it generates a probabilistic estimate of the system's state, often using statistical tools like Bayesian inference to combine model and data uncertainties. This helps quantify the reliability of predictions and allows for a best estimate that minimizes error across both sources.

In this thesis, data assimilation will be applied to the Lorenz 1963, Rössler, and Van der Pol systems. The first two are chaotic systems, while the latter is not. Across both categories, the effectiveness of data assimilation in addressing errors, regardless of a system's complexity, will be demonstrated. These three systems serve as representative examples of more complex unsteady phenomena, such as turbulent flows, highlighting the potential applications of data assimilation techniques in complex real-world scenarios.

Data assimilation involves two types of computations. Those that aim to compute the best estimate by assimilating the current measurement into the current model prediction, and fall within the **Static Data Assimilation** category, and those that aim to predict the present state using the best estimate of the previous system state and its quantified uncertainty. The latter fall within the **Dynamical Data Assimilation** category. Specifically, filtering will be used to estimate the present state of the system, by continuously updating predictions based on new observations and past data. Both categories will be analyzed thoroughly in the following sections.

## 5.2 Notation and Definitions

Consider an unsteady system, the true state of which i.e its state variables, at a fixed point in time, is described by the vector $\vec{u}^{\,t} \in \mathbb{R}^n$, where $n \in \mathbb{N}$ and $t$ stands for *true*. For example, in the case of the unsteady flow around an airfoil, the state vector is $\vec{u} = (\rho, \rho v_x, \rho v_y, E)^T \in \mathbb{R}^n$, $n = 4$. Consider also a model, such as the Navier-Stokes (N-S) equations, the numerical solution of which approximates the true state of the system as $\vec{u}^{\,m} \in \mathbb{R}^n$, where $m$ stands for *model*, along with measurements or observations $\vec{v} \in \mathbb{R}^M$, like those made inside a wind tunnel, $M$ is the number of quantities measured at each measurement. The true state cannot be calculated. Instead its best estimate $\vec{u}^{\,a} \in \mathbb{R}^n$ is computed, where $a$ stands for *assimilated*, is the result of a combination of the information contained in the model estimation (N-S) and the available observations (experimental data). The model equation, that proceeds from time-step $k-1$ to time-step $k$ can be written in the form:

$$\vec{u}_k^{\,m} = M\left(\vec{u}_{k-1}^{\,a}\right) \tag{5.1}$$

where $M(\cdot)$ is the model operator, which propagates the state forward in time. Notice how the input to the model at the time-step $k$ is considered to be the assimilated state at this instant. If no data assimilation has taken place at the previous time-step, the best estimate of the state is:

$$\vec{u}_k^{\,m} = M\left(\vec{u}_{k-1}^{\,m}\right) \tag{5.2}$$

The model is subject to error, thus the true state of the system at time $k$ can be expressed as:

$$\vec{u}_k^{\,t} = M\left(\vec{u}_{k-1}^{\,t}\right) + \vec{e}_{k-1}^{\,m} \tag{5.3}$$

The observations are given as a function of the true state as:

$$\vec{v} = H\left(\vec{u}^{\,t}\right) + \vec{e}^{\,o} \tag{5.4}$$

where $\vec{e}^{\,o}$ is the observation error, and $H(\cdot)$ is the interpolation operator. The latter has a dual purpose; It performs interpolation from the grid nodes to the observation sites, and it also might include transformation of the state variables to observations that might be expressed differently. For example, in case the state variables are conservative and the observations are non-conservative [31](p.11). In general, the observation operator, as well as the model, is non-linear, and this will be the case for this analysis. However, in the simple case where the state is being measured at grid points, $H(\cdot)$ degenerates to the identity operator, and Eq. 5.4 merely becomes:

$$\vec{v} = \vec{u}^{\,t} + \vec{e}^{\,o} \tag{5.5}$$

## 5.3 Errors and Auto-Covariances

The model estimate as well as the assimilated state and the observations are all subject to errors, that are assumed to follow the normal distribution [20]. This assumtion is very realistic, as it has been observed that most quantities measured at experiments follow the normal distribution or approximate it very closely, and even if the do not, they can undergo a mathematical transformation so that they do [30]. Also, for simplicity, and without loss of generality, it is assumed that all components of each state and observation vector share the same variance. The relevant errors are defined as:

| | | |
|---|---|---|
| Model estimate error: | $\vec{e}^{\,m} = \vec{u}^{\,t} - \vec{u}^{\,m}$ | (5.6a) |
| Assimilation error: | $\vec{e}^{\,a} = \vec{u}^{\,t} - \vec{u}^{\,a}$ | (5.6b) |
| Observation error: | $\vec{e}^{\,o} = \vec{v} - H\left(\vec{u}^{\,t}\right)$ | (5.6c) |

It is assumed that the observation error $\vec{e}^{\,o}$ is random, meaning that the observations $\vec{v}$ are subject to unbiased random error. Thus, its mean or expected value is zero. The variance of $\vec{e}^{\,o}$ is constant, equal to $\sigma_o$, meaning that all the $e_i^i \in \vec{e}^{\,o}$ are independent from each other. This is equivalent to:

$$mean(\vec{e}^{\,o}) = E(\vec{e}^{\,o}) = \vec{0} \tag{5.7a}$$
$$cov(\vec{e}^{\,o}) = E\left(\vec{e}^{\,o}\vec{e}^{\,o\,T}\right) = \sigma_o^2\mathbf{I} = \mathbf{C^o} \tag{5.7b}$$
$$\text{or: } \vec{e}^{\,o} \sim \mathcal{N}\left(\vec{0}, \mathbf{C^o}\right) \tag{5.7c}$$

where $\mathbf{C^o}$ is the auto-covariance matrix of $\vec{e}^{\,o}$. Similarly, the model error is assumed to have zero mean and a diagonal auto-covariance matrix:

$$mean(\vec{e}^{\,m}) = E(\vec{e}^{\,m}) = \vec{0} \tag{5.8a}$$
$$cov(\vec{e}^{\,m}) = E\left(\vec{e}^{\,m}\vec{e}^{\,m\,T}\right) = \sigma_m^2\mathbf{I} = \mathbf{C^m} \tag{5.8b}$$
$$\text{or: } \vec{e}^{\,m} \sim \mathcal{N}\left(\vec{0}, \mathbf{C^m}\right) \tag{5.8c}$$

The assimilated state this analysis aims to compute should be unbiased, therefore its mean or expected value should be equal to the true state: $E(\vec{u}^{\,a}) = \vec{u}^{\,t}$, and the assimilation error should have zero mean. Also, the assimilated states should be independent of each other , yielding a diagonal auto-covariance matrix.

$$mean(\vec{e}^{\,a}) = E(\vec{e}^{\,a}) = \vec{0} \tag{5.9a}$$
$$cov(\vec{e}^{\,a}) = E\left(\vec{e}^{\,a}\vec{e}^{\,a\,T}\right) = \sigma_a^2\mathbf{I} = \mathbf{C^a} \tag{5.9b}$$
$$\text{or: } \vec{e}^{\,a} \sim \mathcal{N}\left(\vec{0}, \mathbf{C^a}\right) \tag{5.9c}$$

It should be noted that the Initial Conditions (ICs) are also subject to error, which is considered to be similar to the observation error, because, in most real world applications, ICs result from observations. Hence:

$$mean(\vec{e}^{\,IC}) = E(\vec{e}^{\,IC}) = \vec{0} \tag{5.10a}$$
$$cov(\vec{e}^{\,IC}) = E\left(\vec{e}^{\,IC}\vec{e}^{\,IC\,T}\right) = \sigma_o^2\mathbf{I} = \mathbf{C^o} \tag{5.10b}$$
$$\text{or: } \vec{e}^{\,IC} \sim \mathcal{N}\left(\vec{0}, \mathbf{C^o}\right) \tag{5.10c}$$

Variance $\sigma_o$ is assumed to be known, which is typical procedure followed by experimentalists. On the other hand, $\sigma_m$ stands for the model variance and is usually approximated based on prior empirical information.

## 5.4 The Extended Kalman Filter (EKF)

In data assimilation, the observations usually correspond to a very small subset of the total grid points and time steps. They need to be dynamically incorporated into the simulation process and, in this thesis, the selected method to do so, is the filtering algorithm, specifically the Extended Kalman Filter (EKF) [31]. As the simulation progresses, the model state and the observations (whenever and wherever they are available) are combined to produce the assimilated state which replaces the current state. By keeping track of the accumulated error due to model and observation uncertainties, thus practically utilizing all observations until that instant, the assimilation process extracts information from the model and the measurements with mathematically proven optimality. The assimilated state is assumed to be equal to the model estimate plus a correction term, weighed by an optimally selected relaxation coefficient matrix $\mathbf{K}$ a.k.a the Extended Kalman Filter (EKF). The correction term consists of the difference of the observation and the model prediction, and, depending on the case, the latter should be brought at the temporal or spatial location of the observation, as well as the variable type, using the $H(\cdot)$ operator. This is expressed as:

$$\vec{u}^{\,a} = \vec{u}^{\,m} + \mathbf{K}\left(\vec{v} - H\left(\vec{u}^{\,m}\right)\right) \tag{5.11}$$

The goal of this analysis is to calculate $\mathbf{K}$, so that the following two conditions are satisfied:

- The assimilated state error $\vec{e}^{\,a}$ is unbiased (meaning that $E(\vec{u}^{\,a}) = \vec{u}^{\,t}$, or $E\left(\vec{e}^{\,a}\right) = \mathbf{0}$).

- The assimilated state error variance is the minimum among all other estimates [11].

The first condition is already satisfied, as is proven below:

$$\begin{aligned}
E\left(\vec{e}^{\,a}\right) &= -E\left(\vec{e}^{\,a}\right) = E\left(-\vec{e}^{\,a}\right) \\
&= E\left(\vec{u}^{\,a} - \vec{u}^{\,t}\right) \\
&= E\left(\vec{u}^{\,m} + \mathbf{K}\left(\vec{v} - H\left(\vec{u}^{\,m}\right)\right) - \left(\vec{u}^{\,m} + \vec{e}^{\,m}\right)\right) \\
&= E\left(\vec{u}^{\,m} - \vec{u}^{\,t}\right) + \mathbf{K}E\left(\vec{v}\right) - \mathbf{K}E\left(H\left(\vec{u}^{\,m}\right)\right) \\
&= E\left(\vec{e}^{\,m}\right) + \mathbf{K}E\left(\vec{e}^{\,o} + H\left(\vec{u}^{\,t}\right) - \mathbf{K}H\left(\vec{u}^{\,m}\right)\right. \\
&= \mathbf{0} - \mathbf{K}H\left(\vec{u}^{\,m}\right) + \mathbf{K}E\left(\vec{e}^{\,o}\right) + \mathbf{K}E\left(H\left(\vec{u}^{\,t}\right)\right) \\
&= -\mathbf{K}H\left(\vec{u}^{\,m}\right) + \mathbf{K}\mathbf{0} + \mathbf{K}E\left(H\left(\vec{u}^{\,t}\right)\right) \\
&= \mathbf{K}E\left(H\left(\vec{u}^{\,t}\right)\right) - \mathbf{K}H\left(\vec{u}^{\,m}\right)
\end{aligned} \tag{5.12}$$

Note that, since the model estimate $\vec{u}^{\,m}$ is a deterministic quantity, and $H(\cdot)$ a deterministic operator, then $H\left(\vec{u}^{\,m}\right)$ is a deterministic quantity, and:

$$E\left(H\left(\vec{u}^{\,m}\right)\right) = H\left(\vec{u}^{\,m}\right) \tag{5.13}$$

$$E\left(\vec{u}^{\,m}\right) = \vec{u}^{\,m} \tag{5.14}$$

To further simplify Eq. 5.12, due to the presence of $E\left(H\left(\vec{u}^{\,t}\right)\right)$, the nonlinear operator $H(\cdot)$ has to be linearized, because the non-linear evolution of the probability distribution $H\left(\vec{u}^{\,t}\right)$ can result in a non-Gaussian output, even though the input is Gaussian [31]. The non-linear operator $H(\cdot)$ can be linearized around $\vec{u}^{\,m}$ as:

$$H\left(\vec{u}^{\,t}\right) = H\left(\vec{u}^{\,m}\right) + \nabla\mathbf{H}|_{\vec{u}^{\,m}}\left(\vec{u}^{\,t} - \vec{u}^{\,m}\right) + O\left(\left(\vec{u}^{\,t} - \vec{u}^{\,m}\right)^2\right) \tag{5.15}$$

where $\nabla\mathbf{H}|_{\vec{u}^{\,m}}$ is the Jacobian of $H(\cdot)$ evaluated at $\vec{u}^{\,m}$:

$$\nabla\mathbf{H}|_{\vec{u}^{\,m}} = \left.\frac{\partial H}{\partial \vec{u}}\right|_{\vec{u}=\vec{u}^{\,m}} \tag{5.16}$$

Thus, Eq. 5.12 becomes:

$$\begin{aligned} &E\left(\vec{e}^{\,a}\right) = \mathbf{K}E\left(H\left(\vec{u}^{\,t}\right)\right) - \mathbf{K}H\left(\vec{u}^{\,m}\right) \;\; = \mathbf{K}H\left(\vec{u}^{\,t}\right) - \mathbf{K}H\left(\vec{u}^{\,m}\right) = \mathbf{0} \Rightarrow \\ &E\left(\vec{e}^{\,a}\right) = \mathbf{0} \end{aligned} \tag{5.17}$$

However, $\mathbf{K}$ is still unknown. In order to compute it, one should take into account the second condition, which states that the assimilated state error $\vec{e}^{\,a}$ needs to have the minimum variance among all other estimates of the true state. The mathematical equivalent of this, is that the sum of the diagonal elements of the error auto-covariance matrix $\mathbf{C^a}$, i.e its trace $tr\left(\mathbf{C^a}\right)$ which consists of the squared sum of the variances of each element of $\vec{e}^{\,a}$, needs to be minimized. The derivation of $\mathbf{K}$ can be found in full detail in D.4, but a shorter version of it is also presented in the following part of the current section. In order to derive $\mathbf{C^a}$, $\vec{e}^{\,a}$ has to be derived first. For the present state, at time $k$, it can be shown that:

$$\vec{e}_k^{\,a} = \left(\mathbf{I} - \mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a} + \vec{e}_{k-1}^{\,m}\right) - \mathbf{K}_k\vec{e}_k^{\,o} \tag{5.18}$$

Having derived $\vec{e}_k^{\,a}$, $\mathbf{C^a}_k = E\left(\vec{e}_k^{\,a}\vec{e}_k^{\,a\,T}\right)$ can also be derived, $\mathbf{K}_k$ to minimize $\|\vec{e}_k^{\,a}\|^2 = \sum_i \left(\vec{e}_{k,i}^{\,a}\right)^2 = tr\left(\mathbf{C^a}_k\right)$, as mentioned previously. As shown in D.4:

$$\mathbf{C_k^a} = \left(\mathbf{I} - \mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)\mathbf{C_k^m}\left(\mathbf{I} - \mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)^T + \mathbf{K}_k\mathbf{C_k^o}\mathbf{K}_k^T \tag{5.19}$$

Satisfying $tr\left(\mathbf{C_k^a}\right) = 0$ yields:

$$\mathbf{K} = \mathbf{C^m}\, \nabla \mathbf{H}|_{\vec{u}\,m}^{T} \left[ \nabla \mathbf{H}|_{\vec{u}\,m}\, \mathbf{C^m}\, \nabla \mathbf{H}|_{\vec{u}\,m}^{T} + \mathbf{C^o} \right]^{-1} \tag{5.20}$$

All quantities in Eq. 5.20 refer to time-step $k$, so the corresponding index is omitted. Therefore, $\mathbf{K}$ depends on $H(\cdot)$, $\mathbf{C^o}$ and $\mathbf{C^m}$, which are known.

In the simple case that $H(\cdot)$ is the identity operator and, as previously explained, Eq. 5.5 holds, Eq. 5.20 becomes:

$$\mathbf{K} = \frac{\sigma_m^2}{\sigma_m^2 + \sigma_o^2}\mathbf{I} \tag{5.21}$$

Similarly, Eq. 5.11 becomes:

$$\vec{u}^{\,a} = \vec{u}^{\,m} + \mathbf{K}\left(\vec{v} - \vec{u}^{\,m}\right) = \vec{u}^{\,m} + \frac{\sigma_m^2}{\sigma_m^2 + \sigma_o^2}\left(\vec{v} - \vec{u}^{\,m}\right) = \frac{\vec{u}^{\,m}/\sigma_m^2 + \vec{v}/\sigma_o^2}{1/\sigma_m^2 + 1/\sigma_o^2} \tag{5.22}$$

Eq. 5.22 suggests that the assimilated state $\vec{u}^{\,a}$ is the weighted (by means of the inverted squared variance) sum of the model estimate $\vec{u}^{\,m}$ and the measurement $\vec{v}$. The assimilated state depends more heavily on the component with lower uncertainty. For instance, the more uncertain the model estimate, the more the assimilated state depends on the observation, and vice-versa.

It is also important to keep track of the assimilated state error covariance at each time-step, so having derived $\mathbf{K}$, $\mathbf{C^a}$ can be expressed as a function of the model estimate error auto-covariance matrix $\mathbf{C^m}$ and $\mathbf{K}$, as shown in D.4. Since all quantities correspond to the same time-step, index $k$ is omitted, similar to Eq. 5.20.

$$\mathbf{C^a} = \left(\mathbf{I} - \mathbf{K}\, \nabla \mathbf{H}|_{\vec{u}\,m}\right) \mathbf{C^m} \tag{5.23}$$

It should be noted that $\mathbf{C^a}$ depends on $\mathbf{C^m}$ and $\mathbf{K}$, which in turn depends on $\mathbf{C^m}$ and $\mathbf{C^o}$. By keeping track of $\mathbf{C^a}$ in every time-step, the propagation of the variance of each component of the assimilated state vector $\vec{u}^{\,a}$ through time is known, which is critical information that quantifies its precision, necessary in many applications like robust design.

In the case where Eq. 5.22 holds, Eq. 5.23 can be written as:

$$\mathbf{C^a} = \left(\mathbf{I} - \mathbf{K}\, \nabla \mathbf{H}|_{\vec{u}\,m}\right) \mathbf{C^m} = \left(\mathbf{I} - \frac{\sigma_m^2}{\sigma_m^2 + \sigma_o^2}\mathbf{II}\right) \sigma_m^2 \mathbf{I} \Rightarrow \mathbf{C^a} = \frac{\sigma_m^2 \sigma_o^2}{\sigma_m^2 + \sigma_o^2}\mathbf{I} \tag{5.24}$$

## 5.5 The Data Assimilation Algorithm

Data assimilation consists of two main processes. The static analysis, and the dynamic analysis. The static one aims to produce the best estimate of the system state by combining the model estimate of that state and an observation, all corresponding to the current moment in simulation time, denoted by the subscript $k$, hence the term "static". The dynamic one aims to compute the estimate of the system state corresponding to the next moment in simulation time, $k + 1$, by utilizing the best estimate of the model state at time $k$, propagating the state forward in time, hence the term "dynamic". The static analysis takes place when there is an available observation, which are usually much less than the total simulation time steps, and consequently is less frequent than the dynamic analysis. Also, since the EKF is a function of the model estimate error auto-covariance, the latter has to be calculated at each time step, regardless of whether static analysis takes place or not. Moreover, keeping track of the error auto-covariance provides insight into the model's accuracy and the potential build-up of error through simulation time steps, which is invaluable information for any engineering application. The data assimilation algorithm is summarized in Algorithm 1:

**Algorithm 1** Extended Kalman Filter (EKF)

***Initialize:*** $\vec{u}_0^{\,m}$, $\mathbf{C_0^m} = \mathbf{C_0^m} = \mathbf{C_0^{IC}}$

**for** $k = 0$ to $N - 1$ **do**

**if** $\vec{v}$ exists **then**

***Static Analysis:***

• $\mathbf{K}_k = \mathbf{C_k^m}\, \nabla\mathbf{H}|^T_{\vec{u}_k^{\,m}} \left[ \nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\, \mathbf{C_k^m}\, \nabla\mathbf{H}|^T_{\vec{u}_k^{\,m}} + \mathbf{C_k^o} \right]^{-1}$

{*Calculate the EKF using Eq. 5.20*}

• $\vec{u}_k^{\,a} = \vec{u}_k^{\,m} + \mathbf{K}_k \left( \vec{v}_k - H\left( \vec{u}_k^{\,m} \right) \right)$

{*Compute the assimilated state using Eq. 5.11*}

• $\mathbf{C_k^a} = \left( \mathbf{I} - \mathbf{K}_k\, \nabla\mathbf{H}|_{\vec{u}_k^{\,m}} \right) \mathbf{C_k^m}$

{*Compute the assimilated state auto-covariance matrix using Eq. 5.19*}

***Dynamic Analysis:***

• $\vec{u}_{k+1}^{\,m} = M\left( \vec{u}_k^{\,a} \right)$

{*Calculate the model estimate for the next time step using Eq. D.6*}

• $\mathbf{C_{k+1}^a} = \nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\, \mathbf{C_k^a}\, \nabla\mathbf{M}|^T_{\vec{u}_{k-1}^{\,a}} + \mathbf{C_k^m}$

{*Estimate the optimal state auto-covariance for the next time step using Eq. D.21*}

**else**

***Dynamic Analysis:***

• $\vec{u}_{k+1}^{\,m} = M\left( \vec{u}_k^{\,m} \right)$

{*Calculate the model estimate for the next time step using Eq. D.6, where the current optimal state is* $\vec{u}_k^{\,m}$ *since no assimilation took place at time* $k$}

• $\mathbf{C_{k+1}^a} = \nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\, \mathbf{C_k^a}\, \nabla\mathbf{M}|^T_{\vec{u}_{k-1}^{\,a}} + \mathbf{C_k^m}$

{*Estimate the assimilated state auto-covariance for the next time step using Eq. D.21, where the optimal state at time* $k$ *is the current model prediction since no assimilation took place at time* $k$}

**end if**

**end for**=0

## 5.6 Demonstration in the Van der Pol System

The data assimilation process is applied to the Van der Pol system using the Extended Kalman Filter. The governing equations are shown in Eq. 3.2, which is equivalent to:

$$\frac{d\vec{u}}{dt} = f\left(\vec{u}\right), \qquad \vec{u}(0) = \vec{u}_0 \tag{5.25}$$

where $f(\cdot)$ is given by Eq. B.1. In order to express Eq. 5.25 in the form of Eq. 5.1, it will be discretized using the Euler method:

$$\vec{u}_{k+1} = \vec{u}_k + \left(t_{k+1} - t_k\right) f\left(\vec{u}_k\right) = F\left(\vec{u}_k\right) \tag{5.26}$$

Eq. 5.26 describes the perfect, error free, model of the discretized Van der Pol system. However, in real world applications, models are subject to error, as previously explained. In order to simulate this, a Gaussian error $\vec{e}_k^{\,m}$ will be artificially introduced into $F()$, in the form of Eq. 5.6b, using an RNG based on the Gaussian (Normal) distribution, with mean value 0 and variance $\sigma_m$. The imperfect model can be expressed as in Eq. 5.1, where:

$$M\left(\vec{u}_k^{\,t}\right) = \vec{u}_k + \left(t_{k+1} - t_k\right) f\left(\vec{u}_k\right) + \vec{e}_k^{\,m} \tag{5.27}$$

The observation equation is given by Eq. 5.4, where $\vec{e}^{\,o}$ is artificially introduced, similarly to $\vec{e}_k^{\,m}$. In this case, for simplicity's sake, the observation operator $H(\cdot)$ is reduced to the identity operator:

$$H\left(\vec{u}\right) = \vec{u}, \quad H : \mathbb{R}^2 \to \mathbb{R}^2 \tag{5.28}$$

which means that the observations are directly of the system state itself, and taken at the simulation nodes, but with Gaussian error, as previously discussed. Hence, the Jacobian of the interpolation operator is constant and equal to the identity matrix:

$$\nabla \mathbf{H}|_{\vec{u}_k^{\,m}} = \frac{\partial H}{\partial \vec{u}} = \mathbf{I} \tag{5.29}$$

Thus, the EKF is constant too, and is given by Eq. 5.21. The assimilated state and its error auto-covariance matrix are given by Eqs. 5.22 and 5.23, respectively. The Jacobian of $M$ evaluated at $\vec{u} = \vec{u}_k^{\,a}$ is:

$$\begin{aligned} \nabla \mathbf{M}|_{\vec{u}_{k-1}^{\,a}} &= \frac{\partial M}{\partial \vec{u}_k^{\,a}} = \frac{\partial}{\partial \vec{u}_k^{\,a}} \left[\vec{u}_k^{\,a} + \left(t_{k+1} - t_k\right) f\left(\vec{u}_k^{\,a}\right) + \vec{e}_k^{\,m}\right] \Rightarrow \\ \nabla \mathbf{M}|_{\vec{u}_{k-1}^{\,a}} &= \mathbf{I} + \left(t_{k+1} - t_k\right) \nabla \mathbf{f} \end{aligned} \tag{5.30}$$

where $\nabla \mathbf{f}$ is given by Eq. B.2.

The initial conditions used are $\vec{u}_0 = [0.1,\ 0.1]$, with $N = 50001$ time steps, each of size $dt = 0.001$ time units, so $T = (N-1)dt = 50$ time units. Also, $b = 2$. The true state was evaluated via the Euler method at all time steps in order to provide reference and a basis upon which the observation database was assembled. The latter consists of samples measured every $N_1$ time steps and infused with Gaussian error whose variance is $\sigma_o$. The model error variance is $\sigma_m$. It should be noted that the magnitude of the variances should be selected considering the magnitude of the state.

Fig. 5.1 (Upper) shows the true state (in black), the observation points (in red dots), and the assimilated state (in blue), for $N_1 = 100$, $\sigma_m = 0.01$ and $\sigma_o = 0.05$. In order to visualize the observations' importance in balancing out the model error, Fig. 5.1 (Lower) shows the true state (in black), the observation points (in red dots) and the assimilated state (in blue), where the parameters are kept the same except for $\sigma_m$ which was reduced to half of its original value, so $\sigma_m = 0.005$, and data was used only for the first half of the simulation, i.e for up to 25 time units. After that, the assimilated state corresponds to the model state. It is clear that without the observations, the model error leads to a completely erroneous state, showcasing the importance of data assimilation even with very small model error. It should be noted that the Van der Pol system is not considered chaotic, but it is strongly non-linear. This indicates that even in cases where the system is less sensitive to error than a chaotic one would be, the model error very quickly renders the results unusable, and data assimilation is vital in order to capture the state precisely enough for any kind of analysis.

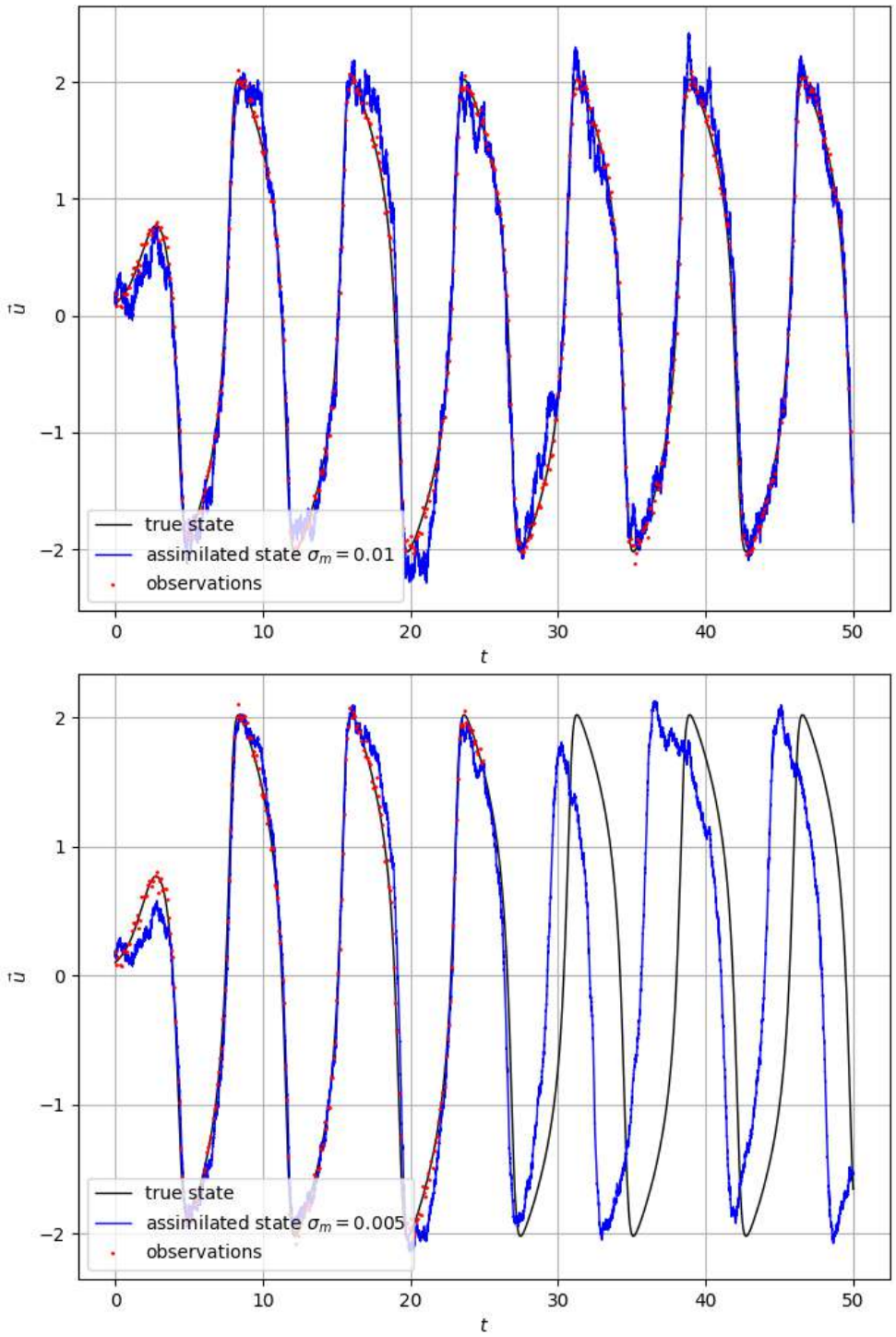



Figure 5.1: *Van der Pol* **Upper**: The true state (in black), the assimilated state (in blue) and the observations (in red dots) for the x-component of the Van der Pol system solution, for four parameter values mentioned in the text. **Lower**: The true state (in black), the assimilated state (in blue) and the observations (in red dots) for the x-component of the Van der Pol system solution, where for time larger than 25 time units, the assimilated state is reduced to the model state, and no observations are used. The parameters were kept the same except that $\sigma_m = 0.005$.

## 5.7 Demonstration in the Lorenz '63 System

The data assimilation algorithm using the EKF is also applied to the Lorenz '63 system, defined as in Eq. 2.1 Similarly to the Van der Pol case study, the ground truth is given by Eq. 5.26, where $f(\cdot)$ is given by Eq. 2.15. The model operator is given by Eq. 5.27, and its Jacobian by Eq. 5.30, where $\nabla \mathbf{f}$ is given by Eq. A.19. The observation operator is the identity operator in this case as well, and the expressions of the assimilated state, the EKF and the auto-covariant matrices are kept the same too, with the difference that vectors are now of length 3 instead of 2, and matrices are of shape $3 \times 3$ instead of $2 \times 2$.

The initial conditions used are $\vec{u}_0 = [1,\ 2,\ 30]$, with $N = 20001$ time steps, each of size $dt = 0.001$ time units, so $T = (N-1)\dot{dt} = 20$ time units. Also, $\sigma = 10$, $\beta = 8/3 \approxeq 2.6667$, $\rho = 28$. The true state was evaluated via the Euler method at all time steps in order to provide reference and a basis upon which the observation database was assembled. The latter consists of 100 samples measured every $N_1$ time steps and infused with Gaussian error with variance equal to $\sigma_o$. The model error variance is $\sigma_m$.

In Fig. 5.4, the true state is plotted along with the assimilated states and the model state which consists of the (erroneous) model predictions without any data assimilation. Clearly, the model fails to estimate the true state quite early in terms of simulation time, while on the other hand, the assimilated state for the most part succeeds to capture its characteristics ('peaks' and 'valleys'). This showcases the impact that data assimilation can have in trying to capture complex, sometimes chaotic, unsteady phenomena.

In Fig. 5.2, the true states were plotted for all time steps (i.e the true trajectory) in black, and the error variances were given the values $\sigma_m = 0.1$, $\sigma_o = 0.5$. Observations were used every $N_1 = 200$ time steps. The x-component of the true state is plotted in black, the assimilated state in blue and the data points are marked by red dots. The process is repeated for eight different Random Number Generator (RNG) seeds, while all other parameters are held constant, in order to examine the effect of randomness on the data assimilation process. The varying precision of the results is a testament to the dependence of the process on the stochastic factor.

In order to examine the effect of the observation error variance $\sigma_o$ on the precision of the assimilated state, three values were tested, and the results are plotted on Fig. 5.3. Once again, the other parameters are held constant and the assimilated states are marked by blue, green and magenta for $\sigma_o = 0.2,\ 0.5,\ 0.7$ respectively. The blue line, i.e the assimilated state for $\sigma_o = 0.2$ stands out as the most precise one, as one can see it has captured the peak at around $t = 8$ and 16 time units, contrary to the other two. This is expected, as it shows that the smaller the observation error covariance, the more precise the assimilated state will be. It should be noted that at some regions of the trajectory, the other two assimilated states are more precise,

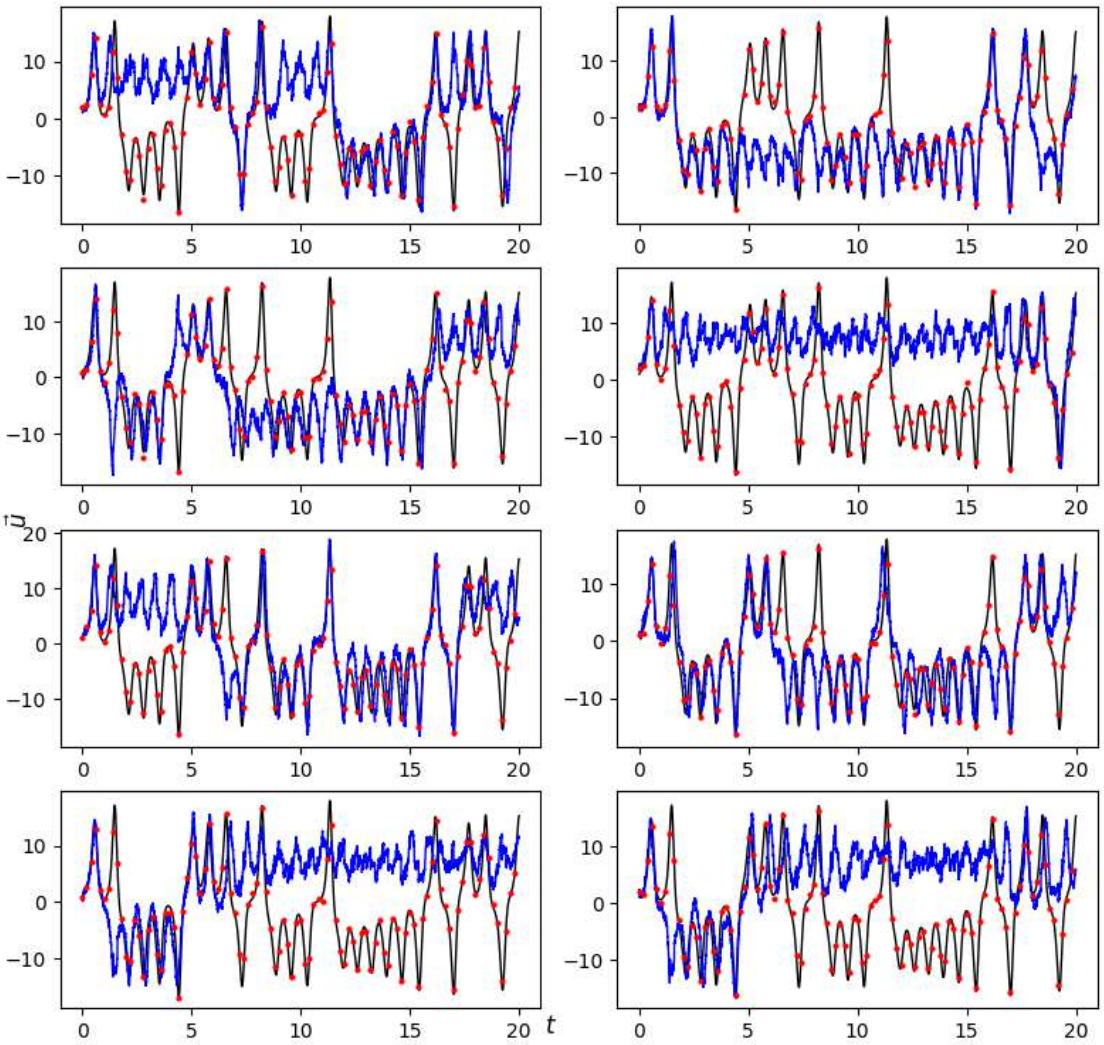



Figure 5.2: *Lorenz '63* The true state (in black), the assimilated state (in blue) and the observations (in red dots) are plotted for eight different RNG seeds, showcasing the randomness of data assimilation. The parameters used are mentioned in the text.

like at around $t = 11$, but this is significantly rarer and is attributed to randomness.

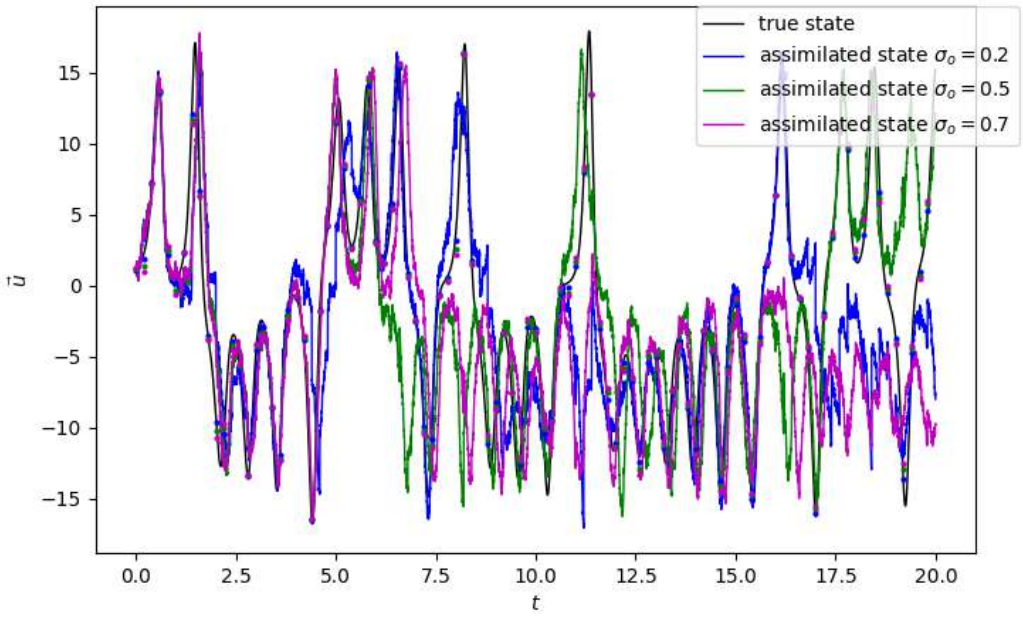



Figure 5.3: *Lorenz '63* The true state (in black), the observations and the assimilated states for $\sigma_o = 0.2,\ 0.5,\ 0.7$. The parameters used are mentioned in the text.

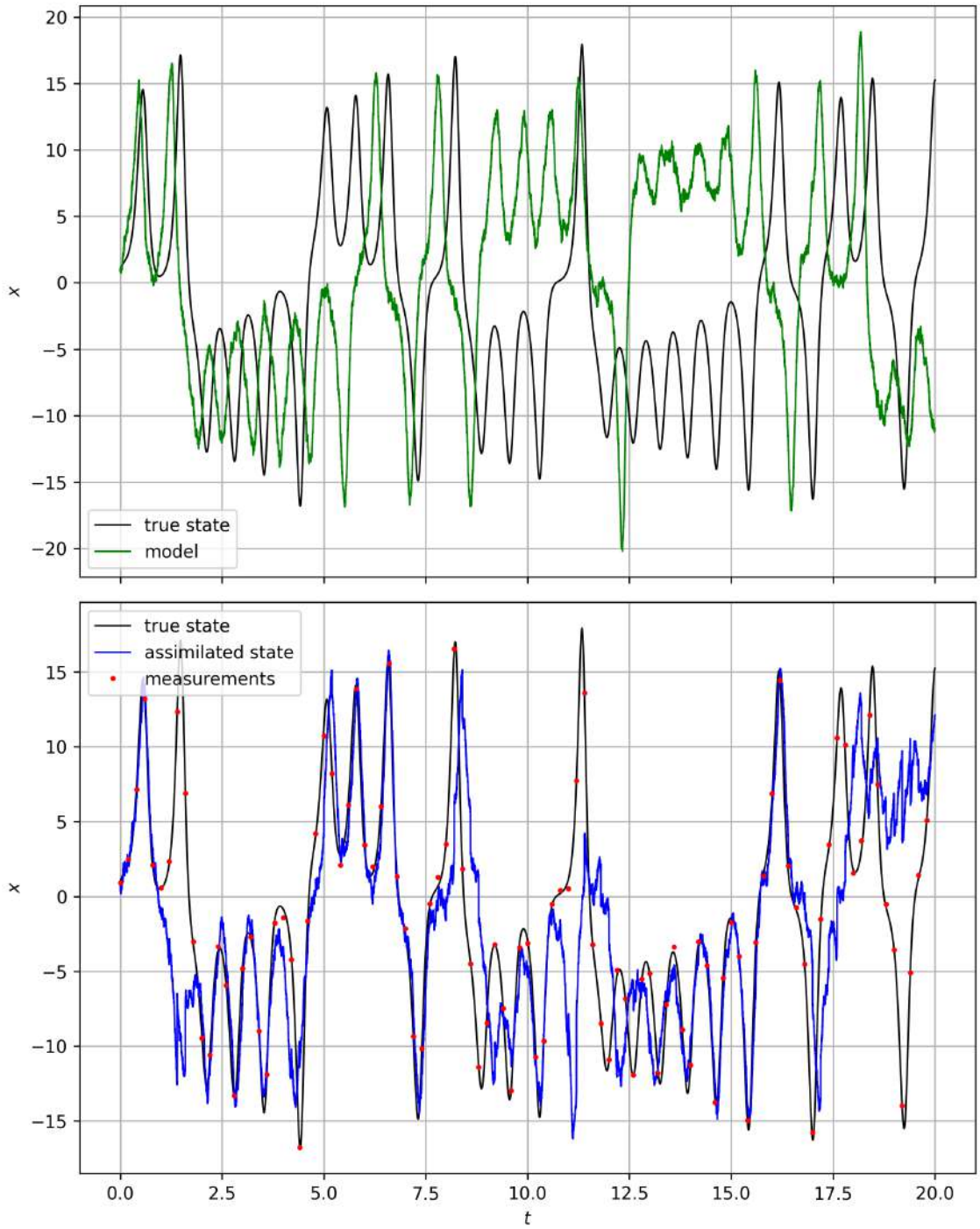



Figure 5.4: *Lorenz '63* The true state (in black), the model state (in green), the measurements (in red) and the assimilated state (in blue) for $\sigma_o = 0.3$ and $\sigma_m = 0.1$. The parameters used are mentioned in the text.

## 5.8 Demonstration in the Rössler System

The data assimilation algorithm using the EKF is also applied to the Rössler system, defined as in Eq. 4.1 Similarly to the Lorenz '63 case study, it is expressed as in Eq. 5.26, where $f(\cdot)$ is given by Eq. C.1. The model operator is also given by Eq. 5.28, and its Jacobian by Eq. C.2. The observation operator is the identity operator in this case as well, and the expressions of the assimilated state, the EKF and the auto-covariant matrices are kept the same too.

The initial conditions used are $\vec{u}_0 = [5,\ 5,\ 5]$, with $N = 30001$ time steps, each of size $dt = 0.001$ time units, so $T = (N-1)dt = 30$ time units. Also, $a = 0.2$, $b = 0.2$, $c = 5.7$. The true state was evaluated via the Euler method at all time steps in order to provide reference and a basis upon which the observation database was assembled. The latter consists of samples measured every $N_1$ time steps and infused with Gaussian error whose variance is $\sigma_o$. The model error variance is $\sigma_m$.

In Fig. 5.5, the true state were plotted for all time steps (i.e the true trajectory) along with the assimilated state. The error variances were set to $\sigma_m = 0.01$, $\sigma_o = 0.3$ and observations were used every $N_1 = 50$ time steps. The z-component of the true state is plotted in black, the assimilated state in blue and the data points are marked by red dots. The process is repeated for four different RNG seeds, while all other parameters are held constant, in order to examine the effect of randomness on the data assimilation process. Also, the peaks are captured very well, even with limited data at the region on and around them, which showcases the precision of the data assimilation process.

Fig. 5.6 (Upper) shows the x-component of true state (in black), the corresponding assimilated and model states in (in blue and green respectively) and the measurements (in red). The model state completely fails to estimate the true dynamics, but the assimilated state manages to almost coincide with it. It should be noted that the Rössler system is significantly more chaotic than the Lorenz '63 system, thus more sensitive in discrepancies due to error, so in order for decently accurate results to be produced, the variances have to be lower and the measurements more frequent.

In order to examine the effect of the model error variance $\sigma_m$ on the assimilated state accuracy, in Fig. 5.6 (Lower), three different assimilated states can be seen that correspond to three different values of the model error variance, namely $\sigma_m = 0.01$ in blue, $\sigma_m = 0.02$ in green and $\sigma_m = 0.03$ in magenta, while the other parameters are held constant and equal to the values mentioned in the previous paragraph. All three variances in the observation error result to assimilated states very close to the true state. There are a few mostly imperceptible fluctuations for $\sigma_m = 0.02$ and 0.03 in the assimilated state, however, that are not present for $\sigma_m = 0.01$. This is natural, since the measurements are very precise in the latter case. It should be noted that both the Rössler and the Lorenz '63 systems, being chaotic, are extremely

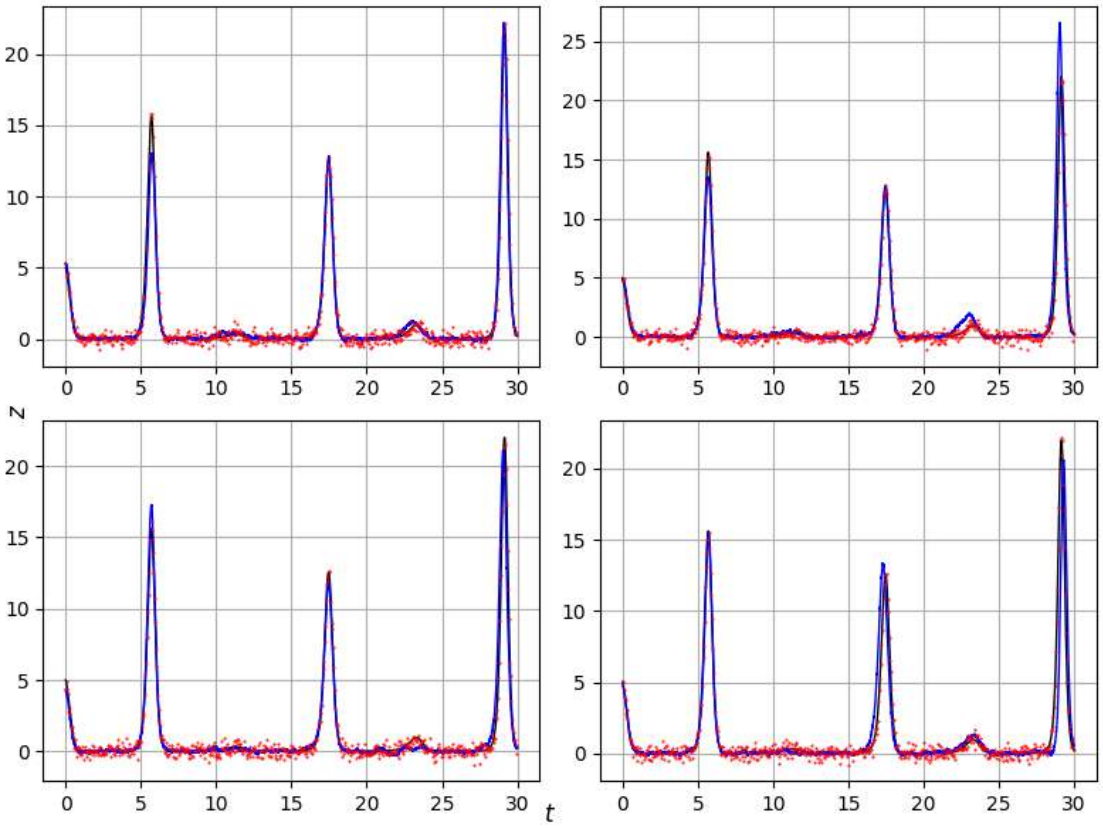



Figure 5.5: *Rössler* The true state (in black), the assimilated state (in blue) and the observations (in red dots) for the solution's z-component, for four different RNG seeds.

sensitive to initial conditions, and the results of both cases showcase the potential of data assimilation to capture complicated dynamics with reasonable cost because due to data collection. Also, the variances selected for the Lorenz 1963 and the Rössler problems differ by one order of magnitude. The reason for that lies on the magnitude of the system state, which is approximately one order smaller in the Rössler system, thus the variance has to be adjusted.

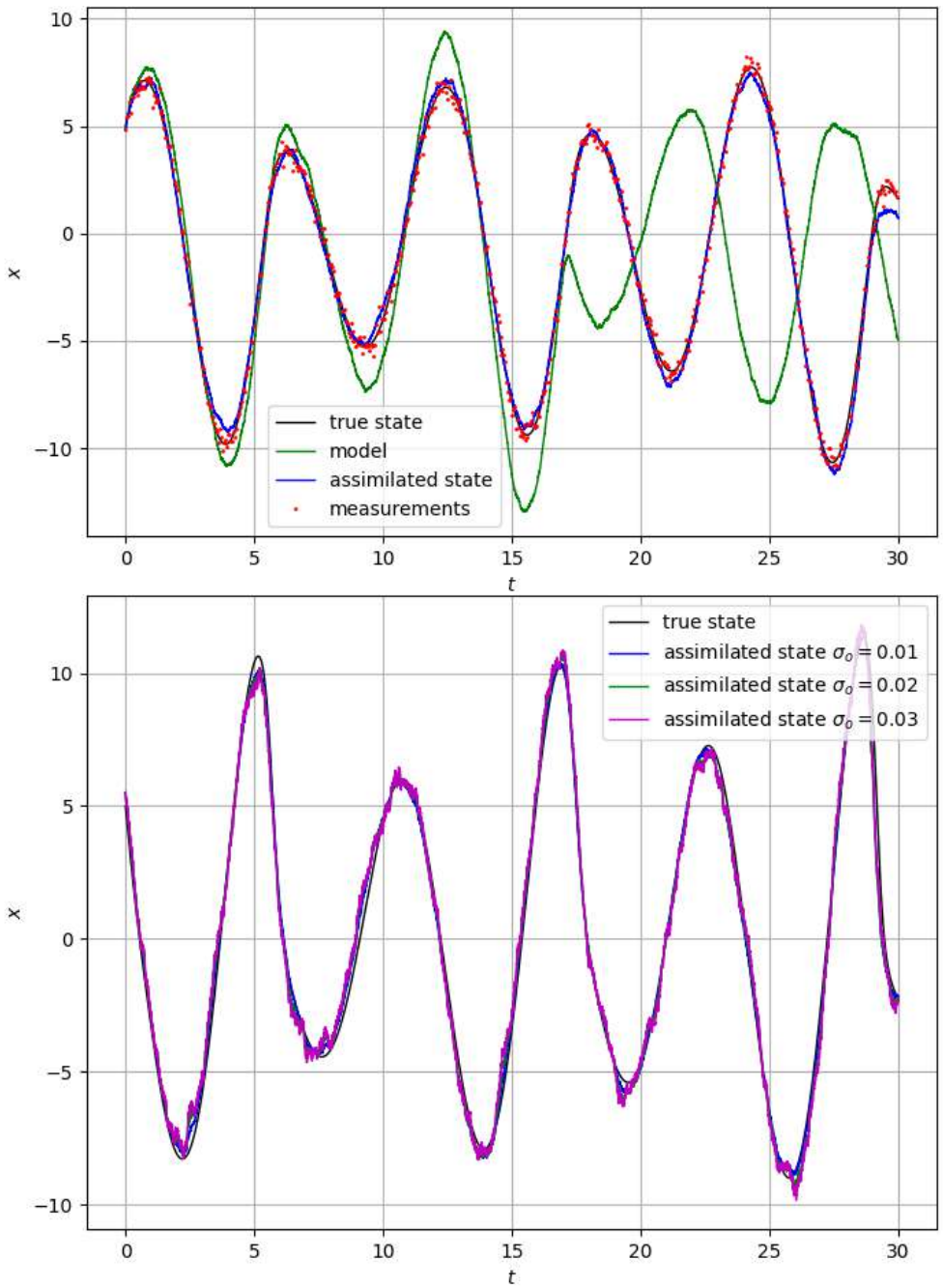



Figure 5.6: *Rössler* **Upper**: The true state (in black), the assimilated state (in blue), the model state (in green) and the observations (in red dots) for the solution's x-component are plotted for $\sigma_m = 0.01$ and $\sigma_0 = 0.3$. **Lower**: The true state (in black), the assimilated state (in blue) and the observations (in red dots) for the z-component of the Rössler system solution are plotted for three different observation error variances, namely $\sigma_m = 0.01$ in blue, $\sigma_m = 0.02$ in green and $\sigma_m = 0.03$ in magenta.

# Chapter 6

# Flow Solution Using Physics-Informed Neural Networks (PINNs)

## 6.1 PINNs as Flow Solvers

Physics Informed Neural Networks or PINNs are a class of neural networks that solve differential equations by minimizing a loss function that consists of the residuals of those equations and the boundary conditions that accompany them. The architecture of the PINNs **??** used in this thesis is based on the Deep Neural Network (DNN) model type, which consists of feed-forward input, hidden and output layers. Contrary to traditional numerical methods which rely on discrete equations and domain, PINNs are analytical models and therefore continuous. This implies that they can directly output field values at any point within that domain without the need for interpolation. Moreover, the derivative values that are needed for the residuals' computation are computed using automatic differentiation, which, as an exact and continuous process, eradicates the need for discretization schemes. However, PINNs have certain limitations due to the significant amount of calculations needed **??**, as stability issues that often arise in some cases. In this thesis, PINN solvers are developed for two applications. The first one is an optimization case, where the primal problem consists of a steady, quasi-1D, incompressible and inviscid flow through a duct of varying cross-sectional area. The objective function is minimized when a given (target) pressure field is achieved, as in an inverse design problem. which relies on the duct shape that in turn is described by the design vector. In this case,

a distinct PINN is developed for each problem: one for the primal and another for the adjoint problem. The adjoint equations are derived using the continuous adjoint method, which is necessary, as the adjoint PINN solves continuous equations in a continuous domain. The desired pressure field is, then obtained successfully at the end of the process. The second case is a 2D steady, laminar flow of an incompressible fluid through a duct of varying cross-sectional area. A PINN is developed in order to solve the Navier-Stokes equations and the accompanying boundary conditions, and the solution is compared to that produced by in-house GPU-enabled CFD code PUMA, developed and maintained by the PCOpt/NTUA.

## 6.2 PINN Model Architecture

The training configuration used in this thesis is shown in Fig. 6.1. The training dataset simply consists of randomly chosen collocation points inside the domain, on which the PINN solves the equations. In each training step, only one of these collocation points, denoted as $\vec{x}$, is used as input for the PINN. After the forward propagation through the hidden layers, the state vector $\vec{u}$ (i.e. velocity components and pressure) at that node is output. Keep in mind that the input layer is depicted as a single node for simplicity, but it is comprised of as many nodes as the dimension of the position vector $\vec{x}$, i.e. if $\vec{x} = (x, y)^T$ then the input layer has two nodes. Similarly, the output layer is also comprised of as many nodes as the dimension of the state vector $\vec{u}$, i.e. if $\vec{u} = (u, v, p)^T$ then the output layer has three nodes. The derivatives $\frac{\partial \vec{u}}{\partial \vec{x}}$ and $\frac{\partial^2 \vec{u}}{\partial \vec{x}^2}$ are then computed using automatic differentiation. The loss consists of the weighted mean squared sum of the governing equations' residuals evaluated at all nodes, as well as the boundary conditions. The weights of the loss components depend on the specific dynamics and are chosen empirically. In this thesis, for both cases, the equation residuals' weights are $a_i = 1$, and those of the boundary conditions (BCs) are $a_i = 4$. The increased weight on the BCs is introduced in order to tackle the issues caused due to the boundary nodes being a small percentage of those of the inner domain, causing the information they carry, although crucial for the problem physics, to be underrepresented, leading to difficulties in convergence. After computing the loss, automatic differentiation is used to evaluate the gradients of the loss with respect to all trainable parameters (weights and biases).

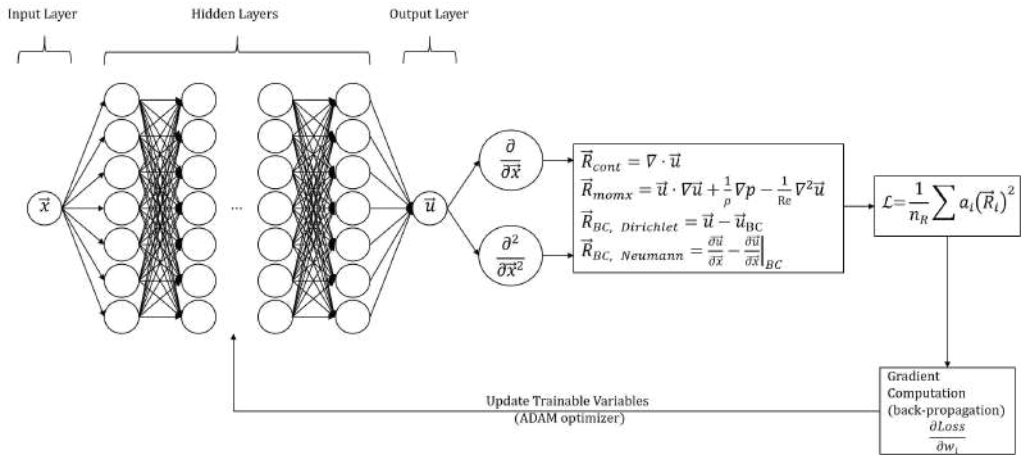



Figure 6.1: A general diagram of the training process of a PINN that solves for the laminar flow of an incompressible fluid.

### 6.2.1 Quasi-1D Flow Case

The first application of the PINN as flow solver is the quasi-1D, steady, incompressible and inviscid flow through a duct with varying cross-sectional area (see Fig.6.2), denoted as $S(x)$, which is given by the following formula:

$$S(x) = b_1\left(x^3 - x\right) + b_2\left(x^2 - x\right) + 0.1x + 0.3 \tag{6.1}$$

where $\vec{b} = (b_1, b_2)^T$ is a parameter vector to be used as a design vector within an optimization loop. It should be noted that the parametrization does not affect the inlet and outlet diameter, which are both constant.

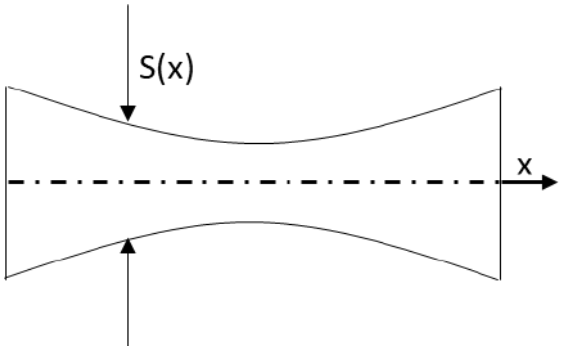



Figure 6.2: The duct used in the quasi-1D flow case.

### Primal Equations

Since the problem is quasi-1D, $\vec{x} = x \in \mathbb{R}^1$, $0 \leq x \leq 1$. The governing equations, i.e. the continuity equation and the momentum conservation, are:

$$\begin{aligned} \frac{d(vS)}{dx} &= 0 \\ \rho v \frac{dv}{dx} + \frac{dp}{dx} &= 0 \end{aligned} \tag{6.2}$$

The state vector is $\vec{u} = (v, p)^T \in \mathbb{R}^2$, where $v$ is the velocity and $p$ the pressure. Since the flow is incompressible, the density remains constant, and for simplicity it is considered $\rho = 1 kg/m^3$. The boundary conditions are:

$$\begin{aligned} u(x = 0) = u_{BC} \quad &= 1\ m/s \\ p(x = 1) = p_{BC} \quad &= 0\ N/m^2 \end{aligned} \tag{6.3}$$

where $p = 0\ N/m^2$ is the reference pressure.

The amount of layers used in the PINN model used to solve Eq. 6.3 was 5, each containing 64 neurons. The input and output layers contained 1 and 2 neurons respectively. For each of the hidden layers, $\tanh(\cdot)$ was used as the activation function, whereas in the output layer, no activation was used. In total, 5000 epochs were performed, and the training duration was 1.3 min. Exponentially decreasing learning rate was used, ranging from 0.001 down to 0.0001 withing the first 2000 epochs, after which it remained constant. The amount of collocation points was 501, and they were introduced batch-wise to the input layer of the PINN, thus taking advantage of the optimized TensorFlow operations. The resulting flow field, plotted alongside a reference field obtained via numerical solution of Eq. 6.3 is shown in Fig.6.3. The two sets of fields are indistinguishable, showcasing the precision achieved by the PINN.

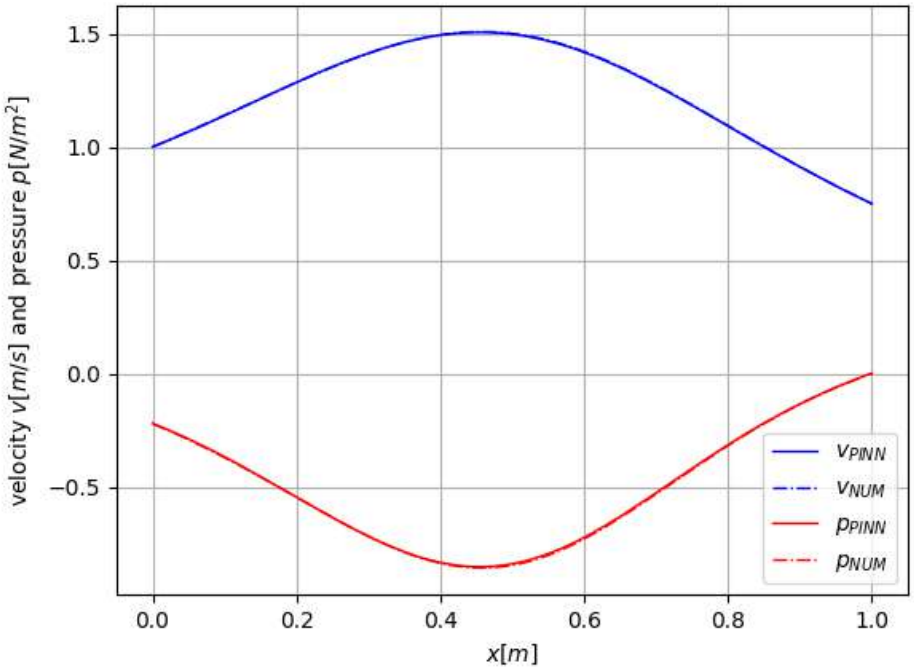



Figure 6.3: **Primal 1D:** Velocity (in blue) and pressure (in red) fields obtained using the PINN (line) compared to numerical solution (dash-dot).

### Adjoint Equations

For the optimization that follows in the next subsection, the objective function is defined as:

$$F(\vec{b}) = \frac{1}{2}\int_0^1 (p - p_{tar})^2 \, dx \tag{6.4}$$

where $p_{tar}$ is a target pressure distribution that corresponds to the target design vector $\vec{b}_{tar} = (0.2, 0.3)^T$. This was selected in order to have an entirely reproducible target for reference purposes. According to the continuous adjoint methodology, the augmented objective function is:

$$F_{aug}(\vec{b}) = F\left(\vec{b}\right) + \int_0^1 v_a \left(\frac{d(vS)}{dx}\right) \, dx + \int_0^1 p_a \left(v\frac{dv}{dx} + \frac{1}{\rho}\frac{dp}{dx}\right) \, dx \tag{6.5}$$

where $v_a$ and $p_a$ are the adjoint velocity and adjoint pressure fields respectively. Differentiating Eq. 6.5 w.r.t. the design vector components $b_n, n = 1, 2$, and since

the operators $\frac{\delta}{\delta b_n}$ and $\frac{d}{dx}$ are interchangeable, results in:

$$\begin{aligned}
\frac{\delta F_{aug}}{\delta b_n} &= \int_0^1 (p - p_{tar}) \frac{\delta p}{\delta b_n} \, dx + \int_0^1 v_a \frac{\delta}{\delta b_n} \left( \frac{d(vS)}{dx} \right) \, dx + \int_0^1 p_a \frac{\delta}{\delta b_n} \left( v \frac{dv}{dx} + \frac{1}{\rho} \frac{dp}{dx} \right) \, dx \\
&= \int_0^1 (p - p_{tar}) \frac{\delta p}{\delta b_n} \, dx + \int_0^1 v_a \frac{d}{dx} \left( \frac{\delta (vS)}{\delta b_n} \right) \, dx + \int_0^1 p_a \left( \frac{\delta}{\delta b_n} \left( v \frac{dv}{dx} \right) + \frac{1}{\rho} \frac{d}{dx} \left( \frac{\delta p}{\delta b_n} \right) \right) \, dx \\
&= \int_0^1 (p - p_{tar}) \frac{\delta p}{\delta b_n} \, dx + \left[ v_a \left( S \frac{\delta v}{\delta b_n} + v \frac{\delta S}{\delta b_n} \right) \right]_0^1 - \int_0^1 \frac{dv_a}{dx} \left( S \frac{\delta v}{\delta b_n} + v \frac{\delta S}{\delta b_n} \right) \, dx \\
&+ \int_0^1 p_a \frac{dv}{dx} \frac{\delta v}{\delta b_n} \, dx + \left[ p_a v \frac{\delta v}{\delta b_n} \right]_0^1 - \int_0^1 \frac{d}{dx} (p_a v) \frac{\delta v}{\delta b_n} \, dx + \frac{1}{\rho} \left[ p_a \frac{\delta p}{\delta b_n} \right]_0^1 - \frac{1}{\rho} \int_0^1 \frac{dp_a}{dx} \frac{\delta p}{\delta b_n} \, dx \\
&= \int_0^1 \left( -S \frac{dv_a}{dx} + p_a \frac{dv}{dx} - v \frac{dp_a}{dx} - p_a \frac{dv}{dx} \right) \frac{\delta v}{\delta b_n} \, dx + \int_0^1 \left( -\frac{1}{\rho} \frac{dp_a}{dx} + (p - p_{tar}) \right) \frac{\delta p}{\delta b_n} \, dx \\
&+ \left[ (v_a s + p_a v) \frac{\delta v}{\delta b_n} \right]_0^1 + \left[ v_a v \frac{\delta s}{\delta b_n} \right]_0^1 + \left[ p_a \frac{\delta p}{\delta b_n} \right]_0^1
\end{aligned} \tag{6.6}$$

The Field Adjoint Equations (FAEs) are determined by setting the multipliers of $\frac{\delta u_i}{\delta b_n}$ to zero continuously throughout the interval $0 < x < 1$, while the Adjoint Boundary Conditions (ABCs) by setting $\frac{\delta u_i}{\delta b_n}$ to zero at the interval boundaries $x = 0$ and $x = 1$. The FAEs and ABCs are:

$$\begin{aligned}
v \frac{dp_a}{dx} + S \frac{dv_a}{dx} &= 0 \\
\frac{1}{\rho} \frac{dp_a}{dx} - (p - p_{tar}) &= 0 \\
v_a|_{x=1} = \left. \frac{-p_a v}{S} \right|_{x=1}, \quad p_a|_{x=0} &= 0
\end{aligned} \tag{6.7}$$

After satisfying the FAEs and ABCs, the only remaining terms on the right-hand-side of Eq. 6.6 constitute the sensitivity derivatives (SDs):

$$\frac{\delta F}{\delta b_n} = - \int_0^1 v \frac{dv_a}{dx} \frac{\delta S}{\delta b_n} \, dx + \left[ v_a v \frac{\delta S}{\delta b_n} \right]_0^1 \tag{6.8}$$

where:

$$\begin{aligned}
\frac{\delta S}{\delta b_1} &= x^3 - x \\
\frac{\delta S}{\delta b_2} &= x^2 - x
\end{aligned} \tag{6.9}$$

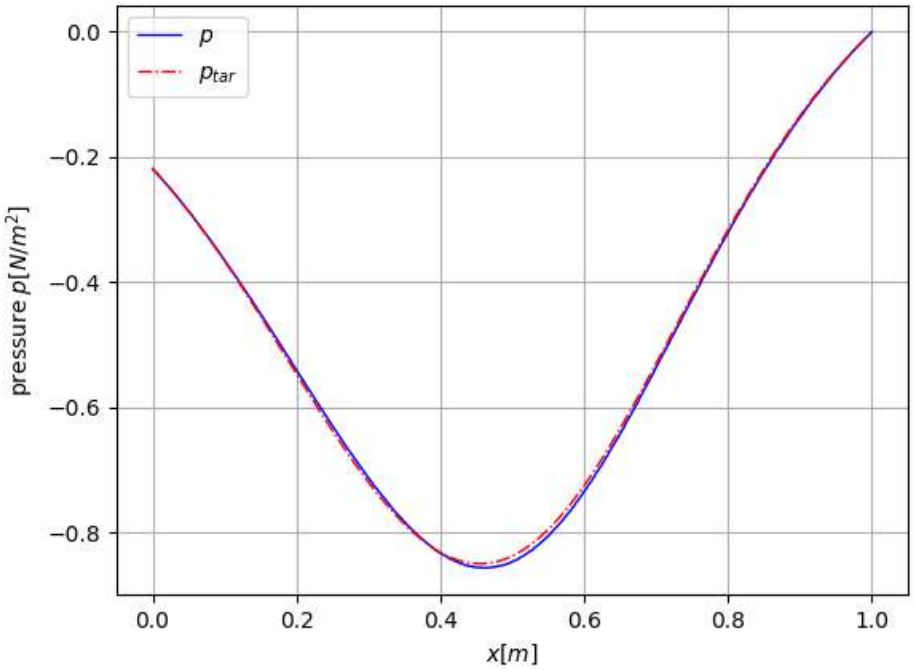



Figure 6.4: The pressure field reached by the optimization process (in blue) vs the target pressure field (in red dash-dot).

#### Optimization Process

The optimization problem, with the objective function of Eq. 6.4, aims to reach a certain pressure distribution that depends on the cross-sectional area of the duct, which in turn is given by Eq. 6.1 and depends on the design vector $\vec{b} = (b_1, b_2)^T$. A simple steepest descent optimizer was used to update the design vector according to:

$$b_n^{new} = b_n^{old} - \eta \frac{\delta F}{\delta b_n}, \quad n = 1, 2, \quad \eta = 0.1 \tag{6.10}$$

After only 13 cycles, where during each of those the primal and adjoint PINNs were re-trained, the objective function reached the lower threshold value chosen to be $F_{threshold} = 5 \cdot 10^{-5}$ and the optimization process was stopped. The resulting pressure distribution is shown in Fig. 6.4, where it is determined that the optimization process was successful.

### 6.2.2 2D Viscous Flow Case

The second application of the PINN as flow solver is the 2D steady viscous laminar incompressible flow through a duct of varying cross-sectional area. The governing

equations are the incompressible Navier-Stokes equations, and the flow is considered to be laminar with $Re = 120$. In vector form they are given by:

$$\begin{aligned} \nabla \cdot \vec{u} &= 0 \\ \vec{u}\nabla\vec{u} - \frac{1}{Re}\nabla^2\vec{u} + \frac{1}{\rho}\nabla p &= 0 \end{aligned} \tag{6.11}$$

where $\vec{u} = (u, v)^T$ and in Einstein notation:

$$\begin{aligned} \frac{\partial u_i}{\partial x_i} &= 0 \\ u_j\frac{\partial u_i}{\partial x_j} - \frac{1}{Re}\frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j}\right) + \frac{1}{\rho}\frac{\partial p}{\partial x_i} &= 0 \end{aligned} \tag{6.12}$$

where:

$$Re = \frac{\rho U L}{\mu} \tag{6.13}$$

and $\rho = 1\ kg/m^3$ for simplicity, $U = 2\ m/s$ is the inlet velocity, $\mu = 0.01\ kg/ms$ the dynamic viscosity and $L = 0.6\,m$ the inlet diameter of the duct. The Boundary Conditions used were:

1. **Inlet:** $u = 1$ and $v = 0$
2. **Outlet:** $p = 0$ (reference pressure) and $\frac{\partial u}{\partial x} = \frac{\partial v}{\partial x} = 0$
3. **Walls:** $u = v = 0$ (no-slip condition)

The hyper-parameters used for the solution using the PINN model, were:

1. 5000 inlet collocation points
2. 5000 outlet collocation points
3. 10000 collocation points for each one of the upper and lower walls
4. 23649 randomly spaced collocation points for the internal region, sampled from a uniform distribution
5. 7 layers of 128 neurons each
6. $\tanh(\cdot)$ activation in all layers, except the output layer which has no activation, a.k.a. "linear" activation
7. initial learning rate 0.002, exponentially decreasing to a minimum of 0.0001 at 2000 epochs, whereon it remains constant

The duct exhibits axial symmetry, and its main dimensions are shown in Fig. 6.5. The collocation points at which the PINN was trained, are shown in Fig. 6.6. The

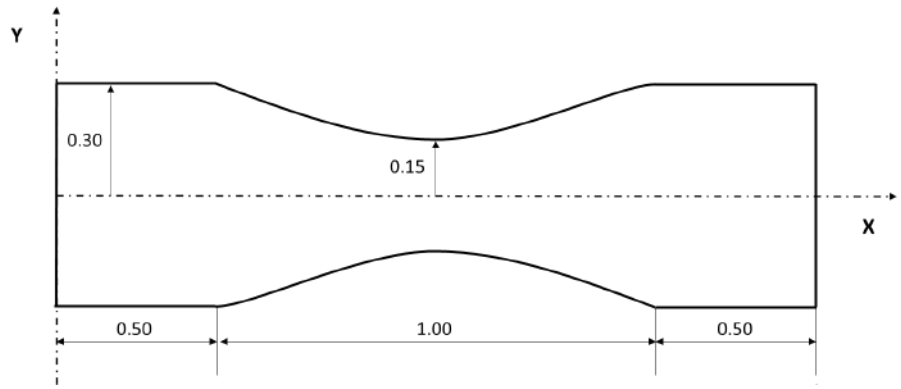



Figure 6.5: The duct geometry used in the 2D flow solution via PINN.

collocation points are twice as dense at a region defined as $-0.1 < y < 0.1$, in order to allow for the flow to fully develop in this region. Also, as it is shown in Fig. 6.7, the region 0.0005 m below the top wall and above the bottom wall, is devoid of any collocation points, the reason being that if there were any, they would interfere with the collocation points placed exactly on the boundary. This is because the boundary collocation points are treated differently from the internal collocation points. Their only contribution to the loss function comes from enforcing the boundary conditions, whereas the internal collocation points contribute through the residuals of the Navier-Stokes equations.

After the PINN solution converged, for demonstration and comparison purposes, the PINN model was called on the points corresponding to a structured H-type grid, on which the flow was solved using the in-house GPU-enabled CFD code PUMA. Figs. 6.8, 6.9 and 6.10 show the horizontal velocity, vertical velocity and pressure fields respectively, visualized using Tecplot 360, with the PINN results on the top, and those from PUMA on the bottom. Fig. 6.11 shows a detail of the vectorised velocity field near the wall, where it can be observed that the boundary layer is very well-formed. In order to further compare the PINN to the PUMA solution, vorticity is plotted along the upper and lower walls, as shown in 6.12. Vorticity, in the general, is defined as:

$$\vec{\omega} = \nabla \times \vec{u} \tag{6.14}$$

In this case, the flow is 2D, so vorticity is a scalar:

$$\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \tag{6.15}$$

Blue dots were used in case of the PINN solution and red dots in case of the PUMA solution. It can be observed that the two vorticity fields practically coincide. Since vorticity is a delicate quantity, consisting of derivatives which tend to amplify numerical errors, those results are indicative of the high quality of the PINN solution.

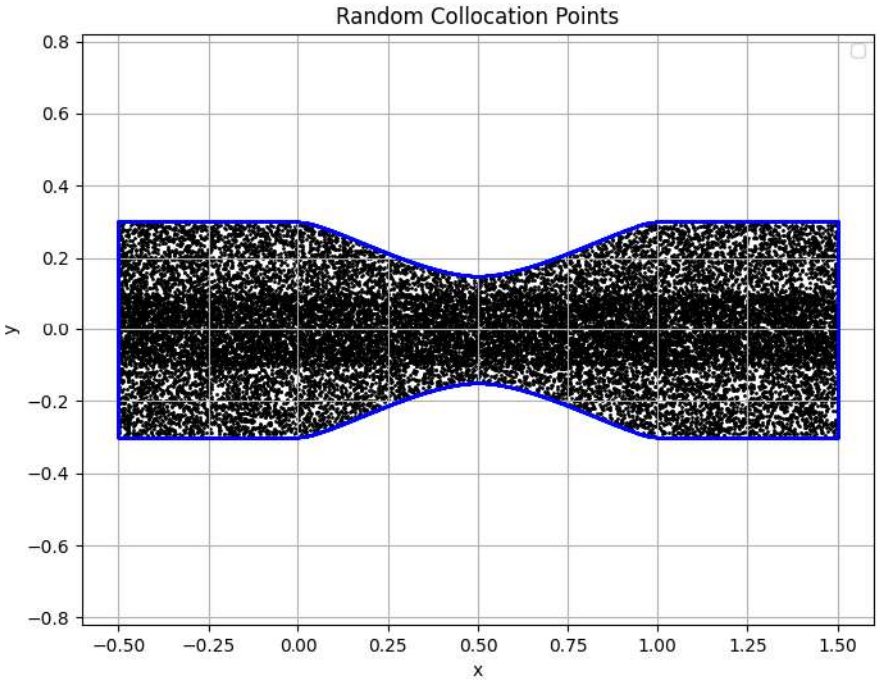



Figure 6.6: The collocation points used in training for the 2D duct flow case. The internal points are marked in black dots, and the boundary points in blue dots.

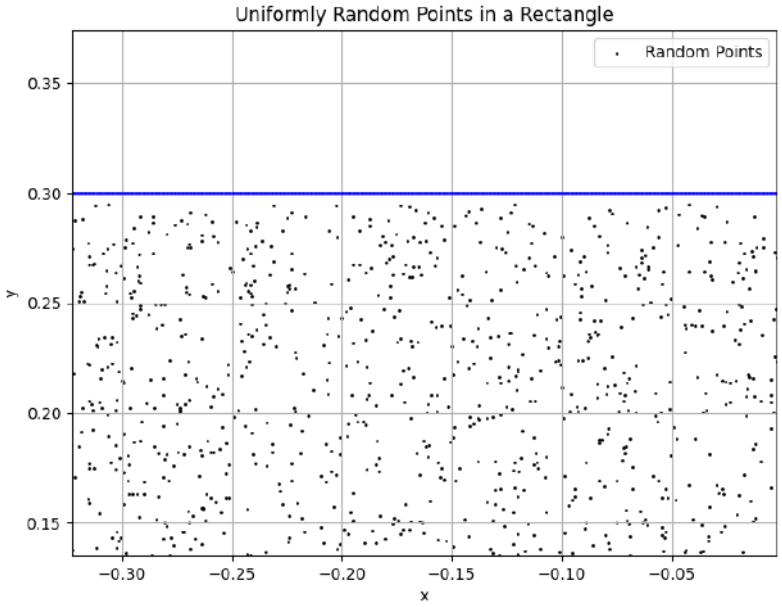



Figure 6.7: Detail near the upper wall, of collocation points used in training for the 2D duct flow case. The internal points are marked in black dots, and the boundary points in blue dots.

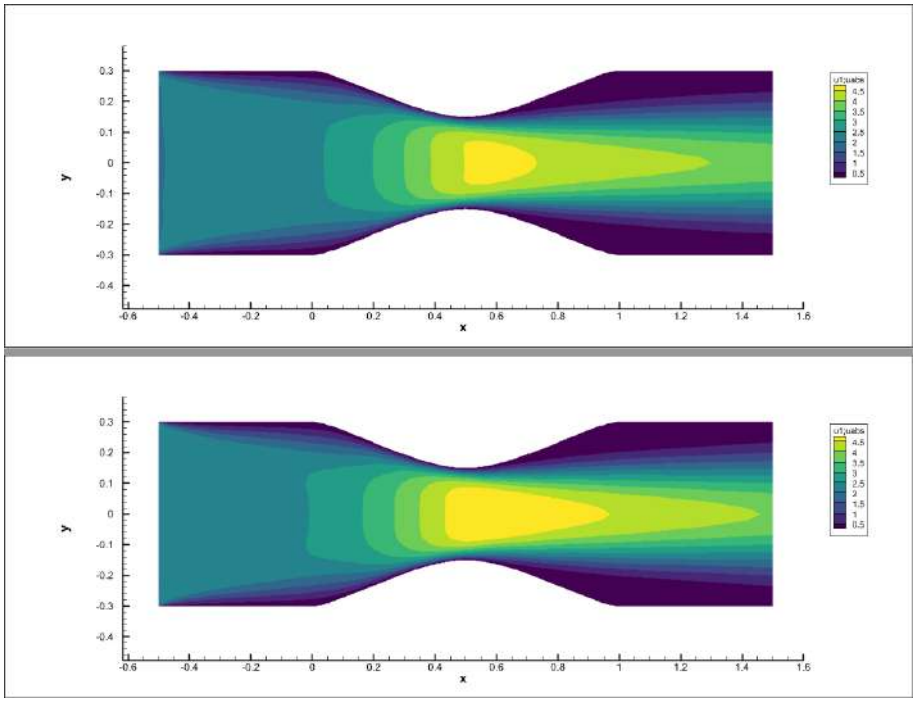


Figure 6.8: Horizontal velocity field of the PINN (upper) and PUMA (lower) solution.

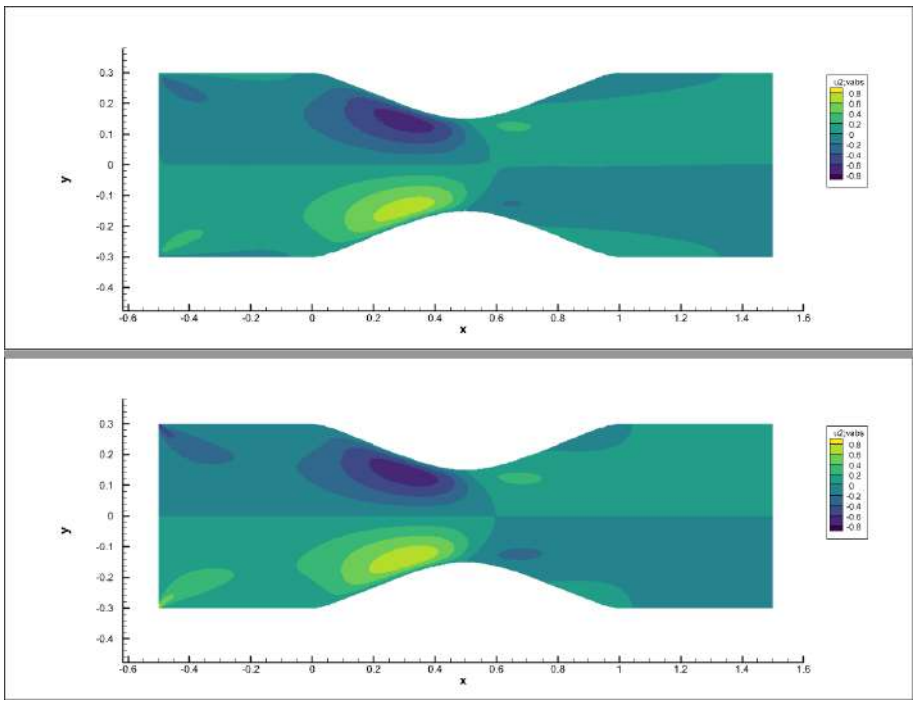


Figure 6.9: Vertical velocity field of the PINN (upper) and PUMA (lower) solution.

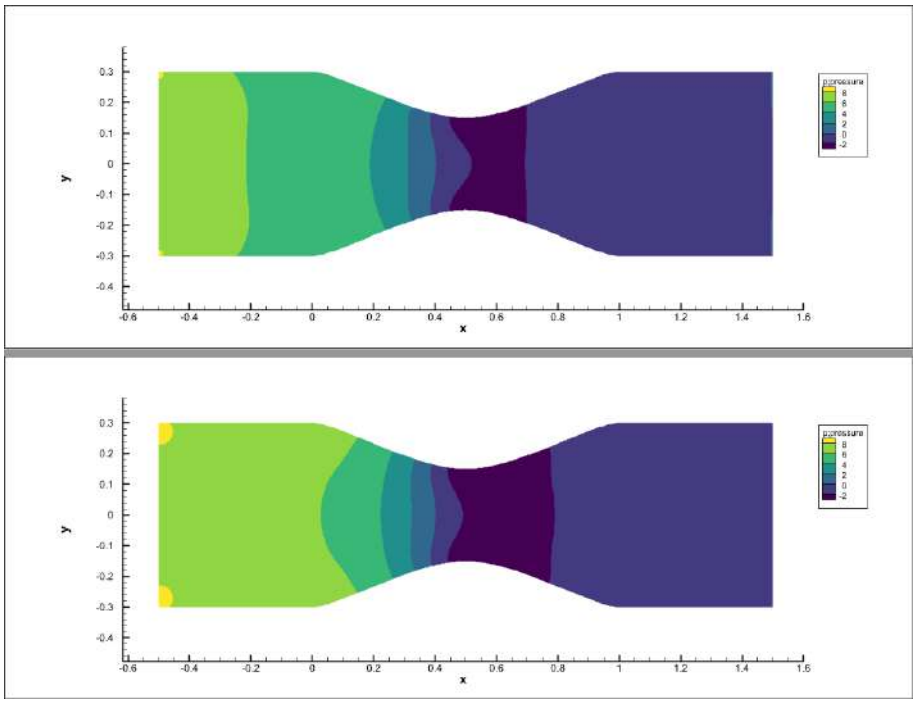


Figure 6.10: Pressure field of the PINN (upper) and PUMA (lower) solution.

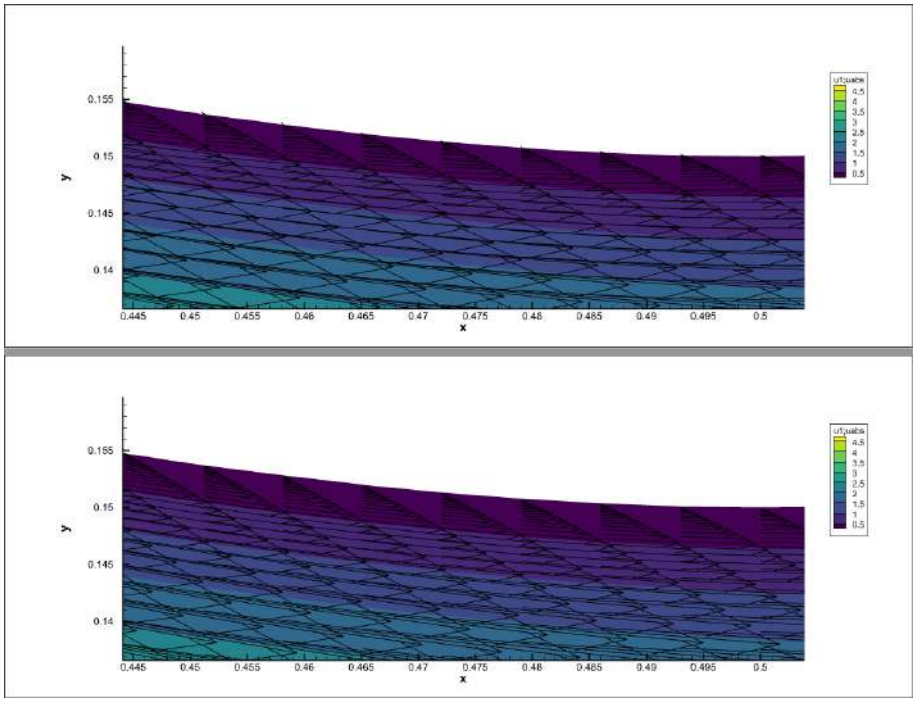



Figure 6.11: Detail of the vectorised boundary layer velocity field of the PINN (upper) and PUMA (lower) solution.

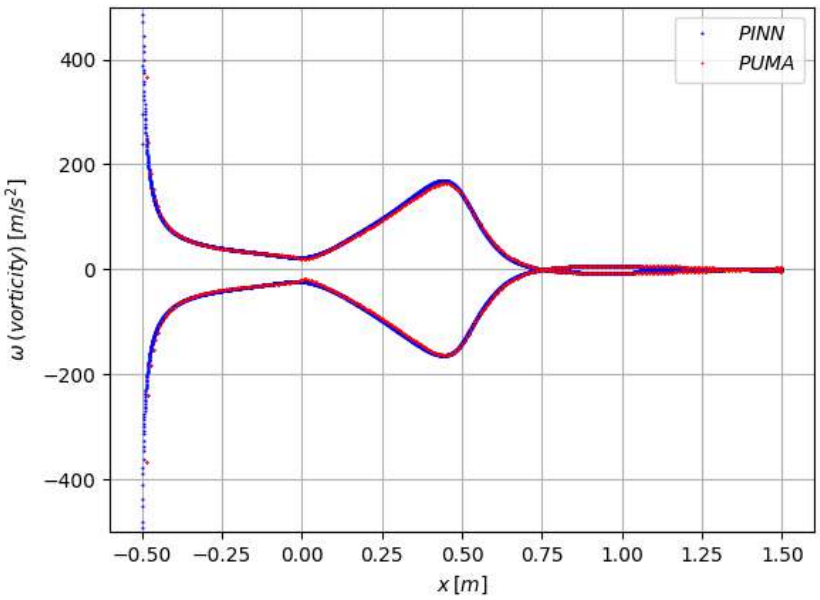



Figure 6.12: Vorticity field of the PINN (upper) and PUMA (lower) solution.

# Chapter 7

# Conclusion

## 7.1 Overview

In this thesis, the failure of the conventional sensitivity analysis methods to produce meaningful sensitivity derivative values when applied to chaotic systems is demonstrated. A solution to this problem is explored, in the form of the Least Squares Shadowing (LSS) algorithm, proposed by Q.Wang et al. [5, 33, 34, 26, 4, 10, 6, 7, 3, 19, 32]. LSS produces a solution for the chaotic problem that corresponds to a perturbed parameter value, which is bound by means of a constrained least squares problem to be close to the trajectory (solution) corresponding to the unperturbed parameter value, for the entire time interval. This perturbed trajectory is guarantied to exist according to proven mathematical theory included in the text, and in effect utilizes the property of ergodicity that characterises many chaotic systems, which states that long-time averaged quantities of those chaotic systems are independent of initial conditions. Thus, the perturbed trajectory is legitimately forced by the minimization problem to accept different initial conditions in order to stay close to the unperturbed trajectory. The comparison of the perturbed and unperturbed trajectories then yields the correct sensitivity derivative value. LSS was first applied on the chaotic system of the Lorenz 1963 equations, and proven to overcome the difficulties faced by the continuous adjoint and finite differences methods. Several parametric studies were performed in order to demonstrate the effect of the size of the time-step, the total integration time and the weight parameter $\alpha$ of the LSS constrained minimization problem on the precision of the sensitivity derivatives. Two other problems were also tested for consistency reasons, the chaotic Rossler system and the Van Der Pol system. Similar parametric studies were performed for those as well.

## 7.2 Conclusions

This process showed that the continuous adjoint method failed completely at each case to produce meaningful derivatives, yielding values that neared the upper limit of double precision numbers. The finite differences method, although it managed to produce meaningful but imprecise sensitivity derivatives, required an extremely large averaging window, which renders it impractical even for small-scale problems such as those explored in this thesis, let alone for orders-of-magnitude more demanding real-life applications. As far as LSS is concerned, it was demonstrated that far less integration time that the FD method was sufficient for high-precision SDs, and the time-step size did not affect the precision of the SD quite as much as in the FD method. The $\alpha$ hyper-parameter also did not greatly affect SD values. These observations overall render LSS a very robust method. However, in most cases the developed DCLSS method appears to yield better results in most cases, requiring less total integration time than classic LSS and producing more accurate SD values due to the consistency of the utilized discretization schemes applied for each of the three LSS equations. Overall, LSS is a robust and reasonably inexpensive method that succeeds in producing precise sensitivity derivatives where other methods either fail altogether or become prohibitively expensive. Additionally, the DCLSS method developed in this thesis is in most cases an improvement over the classic LSS algorithm, providing increased precision at lower computational cost.

Additionally, the Extended Kalman Filter (EKF) method for data assimilation was applied in the Lorenz 1963 and Rossler chaotic systems, as well as the strongly nonlinear but not chaotic Van der Pol system. It successfully managed to overcome the Gaussian error that was introduced in the model and observations, yielding an assimilated state that captured the true dynamics very accurately. It was shown that although the more chaotic a problem is, the less tolerant is the procedure to both the model and the observations' error, great results can be achieved with minimal data, showing potential for larger scale applications such as chaoitic flows.

Finally, Physics-Informed Neural Networks (PINNs) were employed as solvers for steady incompressible flow problems, demonstrating their capability to model fluid dynamics without relying on discrete domain numerical methods. Two applications were explored: a quasi-1D inviscid flow optimization problem and a 2D viscous flow case. For the first application, PINNs were successfully utilized to solve the primal and adjoint equations, allowing for the optimization of the duct shape based on a target pressure field. The second application involved solving the Navier-Stokes equations for a 2D duct flow, comparing the results against the GPU-enabled CFD solver PUMA. The findings showed that PINNs effectively capture the flow characteristics and align well with conventional CFD solutions. However, challenges such as computational cost, training stability, and boundary condition enforcement remain open research areas. This work highlights the potential of PINNs as alternative and/or supportive solvers for fluid flow problems and underscores the need

for further improvements in efficiency and robustness for practical applications.

# Appendix A

# Development of the LSS Equations For the Lorenz '63 system

## A.1 Derivation of the Final Minimization Problem

The minimization problem is defined as:

$$\min_{\vec{u},\eta} \frac{1}{2}\int_0^T \|\vec{u}\,(\tau(t))-\vec{u}_r\|^2+\alpha^2\left(1-\frac{d\tau}{dt}\right)^2 dt, \qquad s.t. \quad \frac{d\vec{u}}{d\tau}=\vec{f}(\vec{u},\rho+\delta\rho), \qquad 0<t<T \tag{A.1}$$

Eq. A.1 ensures that $\vec{u}$ and $\vec{u}_r$, as well as $\tau$ and $t$, remain close to each other for all $0 \leq t \leq T$. Parameter $\alpha$ is selected so that $\|\vec{u}\,(\tau(t)) - \vec{u}_r\|^2$ and $\alpha^2\left(1-\frac{d\tau}{dt}\right)^2$ are of the same order of magnitude. Using Einstein notation, Eq. A.1 can be written as:

$$\min_{\vec{u},\eta} \frac{1}{2}\int_0^T (u_i-u_{r,i})^2+\alpha^2\left(1-\frac{d\tau}{dt}\right)^2 dt, \qquad s.t. \quad \frac{du_{r,i}}{d\tau}=f_i(\vec{u}_r,\rho+\delta\rho), \qquad 0<t<T \tag{A.2}$$

where $\vec{u}_r$ is the solution of:

$$\frac{du_i}{dt}=f_i(\vec{u},\rho) \tag{A.3}$$

and $\vec{u}(t)$ is the shadow trajectory. The time transformation is defined as:

$$\tau(t)=(1+\eta(t)\,\delta\rho)t, \qquad \eta(0)=0 \tag{A.4}$$

In order to write Eq. A.1 in terms of $\frac{\partial u_i}{\partial \rho} \equiv v_i(t)$, it is divided by $\delta\rho^2$. Taking the limit as $\delta\rho \to 0$, it is:

$$\lim_{\delta\rho\to 0} \min_{\vec{u},\eta} \frac{1}{2} \int_0^T \left[ \left( \frac{u_i(\tau(t)) - u_{r,i}(t)}{\delta\rho} \right)^2 + \alpha^2\eta^2 \right] dt = \min_{\vec{v},\eta} \frac{1}{2} \int_0^T \left[ v_i^2 + \alpha^2\eta^2 \right] dt \quad \text{(A.5)}$$

where the following identities were used:

$$\lim_{\delta\rho\to 0} u_i(\tau(t)) = u_{r,i}(t) \qquad \lim_{\delta\rho\to 0} \frac{u_i(\tau(t)) - u_{r,i}(t)}{\delta\rho} = \frac{\partial u_i}{\partial \rho} \equiv v_i(t) \qquad \text{(A.6)}$$

The constraint of the minimization problem is derived in a similar fashion:

$$\begin{aligned}
0 &= \lim_{\delta\rho\to 0} \frac{1}{\delta\rho} \left\{ \left( \frac{du_i}{dt} - f_i(\vec{u}, \rho + \delta\rho) \right) - \left( \frac{du_{r,i}}{dt} - f_i(\vec{u}_r, \rho) \right) \right\} \\
&= \lim_{\delta\rho\to 0} \left\{ \frac{1}{\delta\rho} \left( \frac{1}{1 + \delta\rho\,\eta} \frac{du_i}{dt} - \frac{du_{r,i}}{dt} \right) - \left( \frac{f_i(\vec{u}, \rho + \delta\rho) - f_i(\vec{u}_r, \rho)}{\delta\rho} \right) \right\} \\
&= \lim_{\delta\rho\to 0} \left\{ \frac{1}{1 + \delta\rho\,\eta} \left( \frac{d}{dt} \left( \frac{u_i - u_{r,i}}{\delta\rho} \right) - \eta \frac{du_{r,i}}{dt} \right) \right\} - \frac{\partial f_i}{\partial u_j} v_j - \frac{\partial f_i}{\partial \rho} \qquad \text{(A.7)} \\
&= \frac{dv_i}{dt} - \eta \frac{du_i}{dt} - \frac{\partial f_i}{\partial u_j} v_j - \frac{\partial f_i}{\partial \rho} \\
&= \frac{dv_i}{dt} - \frac{\partial f_i}{\partial u_j} v_j - \frac{\partial f_i}{\partial \rho} - \eta f_i
\end{aligned}$$

By combining equations A.5 and A.7, the minimization problem defined in Eq. A.1, which was cast in terms of $\vec{u}$ and constrained by the primal equation, is transformed to a new minimization problem in terms of $\frac{\partial u_i}{\partial \rho} \equiv v_i(t)$, constrained by the equation of $\vec{v}$. The new minimization problem is essentially the DD equation of $\vec{u}$.

$$\min_{\vec{v},\eta} \frac{1}{2} \int_0^T \left[ v_i^2 + \alpha^2\eta^2 \right] dt, \qquad s.t. \ \frac{dv_i}{dt} = \frac{\partial f_i}{\partial u_j} v_j + \frac{\partial f_i}{\partial \rho} + \eta f_i, \qquad 0 < t < T \quad \text{(A.8)}$$

## A.2 Derivation and Solution of the Karush-Kuhn-Tucker (KKT) equations

The Lagrangian for the minimization problem, in Einstein notation, is defined as:

$$\begin{aligned}\mathcal{L} &= \int_0^T \left[\frac{1}{2}v_i^2 + \frac{1}{2}\alpha^2\eta^2 + w_i\frac{dv_i}{dt} - w_i\frac{\partial f_i}{\partial u_j}v_j - \frac{\partial f_i}{\partial \rho}w_i - \eta w_i f_i\right] dt \\ &= \int_0^T \left[\frac{1}{2}v_i^2 + \frac{1}{2}\alpha^2\eta^2 - v_i\frac{dw_i}{dt} - w_i\frac{\partial f_i}{\partial u_j}v_j - \frac{\partial f_i}{\partial \rho}w_i - \eta w_i f_i\right] dt \\ &+ w_i v_i|_T - w_i v_i|_0\end{aligned} \tag{A.9}$$

for which the KKT equations are:

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial v_i} = 0 = \dfrac{dw_i}{dt} + \dfrac{\partial f_j}{\partial u_i}w_j - v_i \\ \dfrac{\partial \mathcal{L}}{\partial w_i} = 0 = \dfrac{dv_i}{dt} - \dfrac{\partial f_i}{\partial u_j}v_i - \dfrac{\partial f_i}{\partial \rho} - \eta f_i \\ \dfrac{\partial \mathcal{L}}{\partial \eta} = 0 = \eta - \dfrac{1}{\alpha^2}w_i f_i \\ w_i|_{t=0} = w_i|_{t=T} = 0 \end{cases} \tag{A.11}$$

The equivalent of Eq. A.10 in vector notation is:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \vec{v}} &= 0 = \frac{d\vec{w}}{dt} + \left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T \vec{w} - \vec{v} \\ \frac{\partial \mathcal{L}}{\partial \vec{w}} &= 0 = \frac{d\vec{v}}{dt} - \frac{\partial \vec{f}}{\partial \vec{u}}\vec{v} - \frac{\partial \vec{f}}{\partial \rho} - \eta\vec{f} \\ \frac{\partial \mathcal{L}}{\partial \eta} &= 0 = \eta - \frac{1}{\alpha^2}\vec{w}^T\vec{f} \\ \vec{w}(0) &= \vec{w}(T) = 0 \end{aligned} \tag{A.12}$$

In order for the system of Eq. A.12 to properly admit the two BCs, as explained in the text, the KKT equations are combined in a single second order ODE of $\vec{w}$. Differentiating Eq. A.10a w.r.t. $t$ gives:

$$\frac{d^2 w_i}{dt^2} = -\frac{d}{dt}\left(\frac{\partial f_j}{\partial u_i}\right)w_j - \frac{\partial f_i}{\partial u_j}\frac{dw_i}{dt} + \frac{dv_i}{dt} \tag{A.13}$$

Then, replacing $\frac{dv_i}{dt}$ in Eq. A.13 with Eq. A.10b, gives:

$$\frac{d^2w_i}{dt^2} = -\frac{d}{dt}\frac{\partial f_j}{\partial u_i}w_j - \frac{\partial f_i}{\partial u_j}\frac{dw_i}{dt} + \frac{\partial f_i}{\partial u_j}v_i + \frac{\partial f_i}{\partial \rho} + \eta f_i \tag{A.14}$$

After replacing $v_i$ in Eq. A.14 using Eq. A.10a, it becomes:

$$\begin{aligned}&\frac{d^2w_i}{dt^2} = -\frac{d}{dt}\left(\frac{\partial f_j}{\partial u_i}\right)w_j - \frac{\partial f_i}{\partial u_j}\frac{dw_i}{dt} + \frac{\partial f_i}{\partial u_j}\left[\frac{dw_i}{dt} + \frac{\partial f_j}{\partial u_i}w_j\right]w_i + \frac{\partial f_i}{\partial \rho} + \eta f_i \Rightarrow \\ &\frac{d^2w_i}{dt^2} + \left[\frac{\partial f_j}{\partial u_i} - \frac{\partial f_i}{\partial u_j}\right]\frac{dw_i}{dt} + \left[\frac{d}{dt}\left(\frac{\partial f_j}{\partial u_i}\right) - \frac{\partial f_i}{\partial u_j}\frac{\partial f_j}{\partial u_i} - \frac{1}{\alpha^2}f_i f_j\right] - \frac{\partial f_i}{\partial \rho} - \eta f_i = 0\end{aligned} \tag{A.15}$$

to be solved with the two already known boundary conditions:

$$w_i\Big|_{t=0} = w_i\Big|_{t=T} = 0 \tag{A.16}$$

For simplicity, Eq. A.15 can be written:

$$\begin{aligned}&\frac{d^2w_i}{dt^2} + A_i\frac{dw_i}{dt} + B_i w_i - C_i = 0, \qquad w_i|_{t=0} = w_i|_{t=T} = 0 \\ &\text{where: } A_i = \frac{\partial f_j}{\partial u_i} - \frac{\partial f_i}{\partial u_j}, \qquad B_i = \frac{d}{dt}\frac{\partial f_j}{\partial u_i} - \frac{\partial f_i}{\partial u_j}\frac{\partial f_j}{\partial u_i} - \frac{1}{\alpha^2}f_i f_j, \qquad C_i = \frac{\partial f_i}{\partial \rho}\end{aligned} \tag{A.17}$$

The equivalent of Eq. A.17 in vector form is:

$$\begin{aligned}&\frac{d^2\vec{w}}{dt^2} + \mathbf{A}\frac{d\vec{w}}{dt} + \mathbf{B}\vec{w} - \vec{C} = \vec{0}, \qquad \vec{w}|_{t=0} = \vec{w}|_{t=T} = 0 \\ &\text{where: } \mathbf{A} = \left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T - \frac{\partial \vec{f}}{\partial \vec{u}}, \qquad \mathbf{B} = \frac{d}{dt}\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T - \frac{\partial \vec{f}}{\partial \vec{u}}\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T - \frac{1}{\alpha^2}\vec{f}\vec{f}^T, \qquad \vec{C} = \frac{\partial \vec{f}}{\partial \rho}\end{aligned} \tag{A.18}$$

The coefficients $\mathbf{A}$, $\mathbf{B}$ and $\vec{C}$ for the Lorenz '63 equations can be simplified as follows: The Jacobian of the right-hand-side of Eq. 2.1 w.r.t. $\vec{u}$, in the Lorenz '63 problem, is:

$$\frac{\partial \vec{f}}{\partial \vec{u}} = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} \end{pmatrix} = \begin{pmatrix} -\sigma & \sigma & 0 \\ \rho - z & -1 & -x \\ y & x & -\beta \end{pmatrix} \tag{A.19}$$

Thus, the coefficient $\mathbf{A} = \left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T - \frac{\partial \vec{f}}{\partial \vec{u}}$ can be written as:

$$\mathbf{A} = \begin{pmatrix} 0 & \rho - z - \sigma & y \\ \sigma - \rho + z & 0 & 2x \\ -y & -2x & 0 \end{pmatrix} \tag{A.20}$$

The time derivative of the transpose of the Jacobian of the right-hand-side of Eq. 2.1 is:

$$\frac{d}{dt}\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T = \begin{pmatrix} 0 & -\frac{dz}{dt} & \frac{dy}{dt} \\ 0 & 0 & \frac{dx}{dt} \\ 0 & -\frac{dx}{dt} & 0 \end{pmatrix} = \begin{pmatrix} 0 & \beta z - xy & x(\rho - z) - y \\ 0 & 0 & \sigma(y - x) \\ 0 & \sigma(x - y) & 0 \end{pmatrix} \tag{A.21}$$

It is also quite simple to show that:

$$\frac{\partial \vec{f}}{\partial \vec{u}}\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T = \begin{pmatrix} 2\sigma^2 & -\sigma(\rho - z + 1) & \sigma(x - y) \\ -\sigma(\rho - z + 1) & (\rho - z)^2 + 1 + x^2 & y(\rho - z) + x(\beta - 1) \\ \sigma(x - y) & y(\rho - z) + x(\beta - 1) & y^2 + x^2 + \beta^2 \end{pmatrix} \tag{A.22}$$

Matrix $-\frac{1}{\alpha^2}\vec{f}\vec{f}^T$ can be written as:

$$-\frac{1}{\alpha^2}\vec{f}\vec{f}^T = -\frac{1}{\alpha^2}\begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix}\begin{pmatrix} f_1 & f_2 & f_3 \end{pmatrix} = -\frac{1}{\alpha^2}\begin{pmatrix} f_1^2 & f_1 f_2 & f_1 f_3 \\ f_1 f_2 & f_2^2 & f_2 f_3 \\ f_1 f_3 & f_2 f_3 & f_3^2 \end{pmatrix} \tag{A.23}$$

Thus, by taking into account Eqs. A.21, A.21 and A.21 it is:

$$\mathbf{B} = \frac{d}{dt}\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T - \frac{\partial \vec{f}}{\partial \vec{u}}\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T - \frac{1}{\alpha^2}\vec{f}\vec{f}^T \Rightarrow \tag{A.24}$$

$$\mathbf{B} = \begin{pmatrix} -2\sigma^2 - \frac{f_1^2}{\alpha^2} & \beta z - xy + \sigma(\rho - z + 1) - \frac{f_1 f_2}{\alpha^2} & x(\rho z) - y - \frac{f_1 f_3}{\alpha^2} \\ \sigma(\rho - z + 1) - \frac{f_1 f_2}{\alpha^2} & -(\rho - z)^2 - 1 - x^2 - \frac{f_2^2}{\alpha^2} & \sigma(y - x) - y(\rho - z) - x(\beta - 1) - \frac{f_2 f_3}{\alpha^2} \\ -\sigma(x - y) - \frac{f_1 f_3}{\alpha^2} & \sigma(y - x) - y(\rho - z) - x(\beta - 1) - \frac{f_2 f_3}{\alpha^2} & -(x^2 + y^2 + z^2) - \frac{f_3^2}{\alpha^2} \end{pmatrix} \tag{A.25}$$

In this problem, vector $\vec{C}$ is:

$$\vec{C} = \frac{\partial \vec{f}}{\partial \rho} = \begin{pmatrix} \frac{\partial f_1}{\partial \rho} & \frac{\partial f_2}{\partial \rho} & \frac{\partial f_3}{\partial \rho} \end{pmatrix}^T = \begin{pmatrix} 0 & x & 0 \end{pmatrix}^T \tag{A.26}$$

## A.3 Computation of $\vec{v}$ and $\eta$ from $\vec{w}$

After having evaluated $\vec{w}$ for all nodes, in order to find the time-series $\vec{v}$ and $\eta$, Eqs. A.12b and A.12c have to be discretized. For the discretization of the term $\vec{v}$, $\frac{d\vec{w}}{dt}\big|_i$ in Eq. A.12b, forward or backward finite differences can be used. The general form of the discretized equation is:

$$\vec{v}_i = \left.\frac{d\vec{w}}{dt}\right|_i + \left.\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T\right|_i \vec{w}_i \tag{A.27}$$

For forward finite differences, Eq. A.27 is equivalent to:

$$\begin{cases} \vec{v}_i = \dfrac{\vec{w}_{i+1} - \vec{w}_i}{\Delta t} + \begin{pmatrix} -\sigma & \rho - z_i & y_i \\ \sigma & -1 & x_i \\ 0 & -x_i & -\beta \end{pmatrix} \vec{w}_i, & for\ i = 0 \\ \vec{v}_i = \dfrac{\vec{w}_i - \vec{w}_{i-1}}{\Delta t} + \begin{pmatrix} -\sigma & \rho - z_i & y_i \\ \sigma & -1 & x_i \\ 0 & -x_i & -\beta \end{pmatrix} \vec{w}_i, & for\ 1 \leq i \leq N-1 \end{cases} \tag{A.28}$$

For $i = 0$ backward finite differences were used. For backward finite differences, Eq. A.27 is equivalent to:

$$\begin{cases} \vec{v}_i = \dfrac{\vec{w}_{i+1} - \vec{w}_i}{\Delta t} + \begin{pmatrix} -\sigma & \rho - z_i & y_i \\ \sigma & -1 & x_i \\ 0 & -x_i & -\beta \end{pmatrix} \vec{w}_i, & for\ 0 \leq i \leq N-2 \\ \vec{v}_i = \dfrac{\vec{w}_i - \vec{w}_{i-1}}{\Delta t} + \begin{pmatrix} -\sigma & \rho - z_i & y_i \\ \sigma & -1 & x_i \\ 0 & -x_i & -\beta \end{pmatrix} \vec{w}_i, & for\ i = N-1 \end{cases} \tag{A.29}$$

For $i = N - 1$ forward finite differences were used. The discretized form of Eq. A.12c is:

$$\eta_i = \frac{1}{\alpha^2} \vec{w}_i^T \vec{f}_i \tag{A.30}$$

## A.4 Evaluation of the Sensitivity Derivative

Supposing $J = z(t)$, the objective function $F$ for the reference trajectory can be written as:

$$F(\rho) = \left(\frac{1}{T}\int_0^T J(\vec{u}_r, \rho)\, dt\right)^p \tag{A.31}$$

where $p \in \mathbb{I} - \{0\}$. then the objective function for the shadow trajectory corresponding to an input parameter perturbed by $\delta\rho$ is:

$$F(\rho + \delta\rho) = \left( \frac{1}{T'} \int_0^{T'} J(\vec{u}, \rho + \delta\rho)\, d\tau \right)^p \tag{A.32}$$

where $T'$ is the shadow averaging window:

$$T' = \tau(T) = T + \delta\rho\, \zeta, \quad \zeta = \int_0^T \eta\, dt \tag{A.33}$$

The corresponding perturbation of the objective function can be written as:

$$\Delta F = F(\rho+\delta\rho) - F(\rho) = \left( \frac{1}{T'} \int_0^{T'} J(\vec{u}, \rho + \delta\rho)\, d\tau \right)^p - \left( \frac{1}{T} \int_0^T J(\vec{u}, \rho)\, dt \right)^p \tag{A.34}$$

Assuming $\delta\rho \ll 1$ and $\delta\rho^p \cong 0$, for $p \geq 2$:

$$\begin{aligned}
\Delta F &= \left( \frac{1}{T + \delta\rho\, \zeta} \int_0^T J(\vec{u}, \rho + \delta\rho) \frac{d\tau}{dt}\, dt \right)^p - \left( \frac{1}{T} \int_0^T J(\vec{u}_r, \rho)\, dt \right)^p \\
\Rightarrow \delta F &= \frac{1}{T^p + \delta\rho\, p\zeta T^{p-1}} \left\{ \left( \int_0^T J(\vec{u}, \rho + \delta\rho)\, dt \right)^p \right. \\
&\quad + \delta\rho\, p \left( \int_0^T J(\vec{u}, \rho + \delta\rho)\, dt \right)^{p-1} \left( \int_0^T J(\vec{u}, \rho + \delta\rho)\eta\, dt \right)^p \\
&\quad \left. - \left( 1 + \frac{\delta\rho\, p\zeta}{T} \right) \left( \frac{1}{T} \int_0^T J(\vec{u}_r, \rho)\, dt \right)^p \right\}
\end{aligned} \tag{A.35}$$

Dividing by $\delta\rho$ and taking the limit as $\delta\rho \to 0$:

$$\begin{aligned}
\frac{\delta F}{\delta\rho} = \lim_{\delta\rho\to 0} \frac{\Delta F}{\delta\rho} &= \frac{1}{T^p + \delta\rho\, p\zeta T^{p-1}} \left\{ \frac{1}{\delta\rho} \left[ \left( \int_0^T J(\vec{u}, \rho + \delta\rho)\, dt \right)^p - \left( \int_0^T J(\vec{u}, \rho)\, dt \right)^p \right] \right. \\
&\quad + p \left( \int_0^T J(\vec{u}, \rho + \delta\rho)\, dt \right)^{p-1} \left( \int_0^T J(\vec{u}, \rho + \delta\rho)\eta\, dt \right) \\
&\quad \left. - p \left( \int_0^T J(\vec{u}, \rho + \delta\rho)\, dt \right)^{p-1} \frac{\zeta}{T} \left( \int_0^T J(\vec{u}_r, \rho)\, dt \right) \right\}
\end{aligned} \tag{A.36}$$

Using the following identities:

$$\lim_{\delta\rho\to 0} \vec{u}(\tau(t)) = u_r(t), \qquad \lim_{\delta\rho\to 0} \tau(t) = t \tag{A.37}$$

and:

$$\lim_{\delta\rho\to 0} \frac{J(\vec{u},\rho+\delta\rho) - J(\vec{u}_r,\rho)}{\delta\rho} = \langle \frac{\partial J}{\partial \vec{u}_r}, \frac{\partial \vec{u}_r}{\partial \rho} \rangle + \frac{\partial J}{\partial \rho} = \langle \frac{\partial J}{\partial \vec{u}_r}, v \rangle + \frac{\partial J}{\partial \rho} \tag{A.38}$$

$$\begin{aligned} \frac{\delta F}{\delta \rho} = \frac{p}{T^p} \left( \int_0^T J(\vec{u}_r,\rho)\, dt \right)^{p-1} \left\{ \left( \int_0^T \left\langle \frac{\partial J}{\partial \vec{u}_r}, v \right\rangle dt \right) + \left( \int_0^T J(\vec{u}_r,\rho)\, \eta\, dt \right) \right\} \\ - \frac{1}{T} \left( \int_0^T \eta\, dt \right) \left( \int_0^T J(\vec{u}_r,\rho)\, dt \right) + \frac{\partial J}{\partial \rho} \end{aligned} \tag{A.39}$$

In the case of $p = 1$, Eq. A.39 can be written as:

$$\frac{\delta F}{\delta \rho} = \frac{\delta \overline{J}}{\delta \rho} = \overline{\langle \frac{\partial J}{\partial \vec{u}_r}, v \rangle} + \overline{\eta J} - \overline{\eta}\,\overline{J} + \frac{\partial \overline{J}}{\partial \rho} \tag{A.40}$$

The overbar is defined as: $\overline{x} = \frac{1}{T}\int_0^T x\, dt$. For $J = z(t)$, the following simplifications hold:

$$\begin{aligned} \overline{\langle \frac{\partial J}{\partial \vec{u}_r}, v \rangle} &= \frac{1}{T} \int_0^T \left[ \frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}, \frac{\partial z}{\partial z} \right] \vec{v}\, dt \\ &= \frac{1}{T} \int_0^T [0,0,1]\, [v_x, v_y, v_z]^T\, dt \\ &= \frac{1}{T} \int_0^T v_z\, dt \end{aligned} \tag{A.41}$$

$$\frac{\partial \overline{J}}{\partial \rho} = \frac{\partial}{\partial \rho} \left( \frac{1}{T} \int_0^T z\, dt \right) = 0 \tag{A.42}$$

$$\overline{\eta J} = \frac{1}{T} \int_0^T \eta z\, dt \tag{A.43}$$

$$\overline{J} = \frac{1}{T} \int_0^T z\, dt \tag{A.44}$$

Thus, Eq. A.40 can be written as:

$$\frac{\delta F}{\delta \rho} = \frac{1}{T} \int_0^T (v_z + \eta z)\ dt - \frac{1}{T^2} \int_0^T \eta\, dt \int_0^T z\, dt \tag{A.45}$$

## A.5 Derivation of the DCLSS second Order ODE

The equivalent form of Eq. 2.20, using right arrows to denote forward differentiation and left arrows for backward differentiation, is:

$$\begin{aligned}
\frac{\overleftarrow{d\vec{v}}}{dt} &= \frac{\partial \vec{f}}{\partial \vec{u}}\vec{v} + \frac{\partial \vec{f}}{\partial \rho} + \frac{1}{\alpha^2}\vec{f}\vec{f}^T\vec{w} \\
\frac{\overrightarrow{d\vec{w}}}{dt} &= -\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T \vec{w} + \vec{v} \\
\vec{w}(0) &= \vec{w}(T) = 0
\end{aligned} \tag{A.46}$$

Differentiating Eq. 2.29b using backward FD results in:

$$\frac{d^2\vec{w}}{dt^2} = -\frac{\overleftarrow{d}}{dt}\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T \vec{w} - \frac{\partial \vec{f}}{\partial \vec{u}}\frac{\overleftarrow{d\vec{w}}}{dt} + \frac{\overleftarrow{d\vec{v}}}{dt} \tag{A.47}$$

Substituting $\frac{\overleftarrow{d\vec{v}}}{dt}$ using Eq. 2.29a gives:

$$\frac{d^2\vec{w}}{dt^2} = -\frac{\overleftarrow{d}}{dt}\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T \vec{w} - \frac{\partial \vec{f}}{\partial \vec{u}}\frac{\overleftarrow{d\vec{w}}}{dt} + \frac{\partial \vec{f}}{\partial \vec{u}}\vec{v} + \frac{\partial \vec{f}}{\partial \rho} + \frac{1}{\alpha^2}\vec{f}\vec{f}^T\vec{w} \tag{A.48}$$

which can be written as:

$$\frac{d^2\vec{w}}{dt^2} + \left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T \frac{\overleftarrow{d\vec{w}}}{dt} - \frac{\partial \vec{f}}{\partial \vec{u}}\frac{\overrightarrow{d\vec{w}}}{dt} + \left[\frac{\overleftarrow{d}}{dt}\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T - \frac{\partial \vec{f}}{\partial \vec{u}}\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T - \frac{1}{\alpha^2}\vec{f}\vec{f}^T\vec{w}\right]\vec{w} = \frac{\partial \vec{f}}{\partial \rho} \tag{A.49}$$

which is equivalent to:

$$\frac{d^2\vec{w}}{dt^2} + \mathbf{A_1}\frac{\overleftarrow{d\vec{w}}}{dt} - \mathbf{A_2}\frac{\overrightarrow{d\vec{w}}}{dt} + \mathbf{B}\vec{w} = \vec{C} \tag{A.50}$$

where:

$$\begin{aligned}
\mathbf{A_1} &= \left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T \\
\mathbf{A_2} &= -\frac{\partial \vec{f}}{\partial \vec{u}} \\
\mathbf{B} &= \frac{\overleftarrow{d}}{dt}\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T - \frac{\partial \vec{f}}{\partial \vec{u}}\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T - \frac{1}{\alpha^2}\vec{f}\vec{f}^T\vec{w} \\
\vec{C} &= \frac{\partial \vec{f}}{\partial \rho}
\end{aligned} \tag{A.51}$$

and:

$$\begin{aligned}
\left.\frac{d^2\vec{w}}{dt^2}\right|_i &= \frac{\vec{w}_{i-1} - 2\vec{w}_i + \vec{w}_{i+1}}{dt^2} \\
\left.\frac{\overleftarrow{d\vec{w}}}{dt}\right|_i &= \frac{\vec{w}_i - \vec{w}_{i-1}}{dt} \\
\left.\frac{\overrightarrow{d\vec{w}}}{dt}\right|_i &= \frac{\vec{w}_{i+1} - \vec{w}_i}{dt}
\end{aligned} \tag{A.52}$$

# Appendix B

# Derivations in the Van der Pol case

In the Van dre Pol system, the vector of unknowns is bidimentional: $\vec{u}(t) = [x(t)y(t)]^T$. The right-hand-side of Eq. 3.2 is:

$$\vec{f} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} = \begin{pmatrix} y \\ -x + by(1-x^2) \end{pmatrix} \tag{B.1}$$

The Jacobian of Eq. B.1 is:

$$\frac{\partial \vec{f}}{\partial \vec{u}} = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 - 2bxy & b(1-x^2) \end{pmatrix} \tag{B.2}$$

## B.1 Coefficients of the second Order ODE (LSS)

By substituting the previous section's expressions in Eq. A.18, the formula of the coefficient $\mathbf{A}$ is evaluated:

$$\mathbf{A} = \left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T - \frac{\partial \vec{f}}{\partial \vec{u}} = \begin{pmatrix} 0 & -2(1+bxy) \\ 2(1+bxy) & 0 \end{pmatrix} \tag{B.3}$$

For $\mathbf{B}$, it can be shown that:

$$\mathbf{B} = \begin{pmatrix} -1 - \frac{1}{\alpha^2} f_1^2 & -2b\left(y^2 + \left(-x + b * (1-x^2) * y\right) x\right) - b(1-x^2) - \frac{1}{\alpha^2} f_1 f_2 \\ b(1-x^2) - \frac{1}{\alpha^2} f_2 f_1 & -2bxy - (1+2bxy)^2 - b^2 \left(1-x^2\right)^2 - \frac{1}{\alpha^2} f_2^2 \end{pmatrix} \tag{B.4}$$

Also:

$$\vec{C} = \left(0\ (1-x^2)\ y\right)^T \tag{B.5}$$

where $\frac{dx}{dt}$ and $\frac{dy}{dt}$ were substituted using the analytical expression of Eq. 3.2. It is worth noting that the terms $\frac{d^2\vec{w}}{dt^2}$ and $\frac{d\vec{w}}{dt}$ were discretized using $2^{nd}$ order central finite differences.

## B.2 Coefficients of the second Order ODE (DCLSS)

The coefficients $\mathbf{A_1}$ and $\mathbf{A_2}$ of Eq. A.51 are:

$$\mathbf{A_1} = \left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T = \begin{pmatrix} 0 & -1-2bxy \\ 1 & b(1-x^2) \end{pmatrix} \tag{B.6}$$

$$\mathbf{A_2} = -\frac{\partial \vec{f}}{\partial \vec{u}} = -\begin{pmatrix} 0 & 1 \\ -1-2bxy & b(1-x^2) \end{pmatrix} \tag{B.7}$$

It is true that:

$$\frac{\overleftarrow{d}}{dt}\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T = \begin{pmatrix} 0 & -2b\left(\frac{x_i-x_{i-1}}{\Delta t}y_i + \frac{y_i-y_{i-1}}{\Delta t}x_i\right) \\ 0 & -2bx_i\left(\frac{x_i-x_{i-1}}{\Delta t}\right) \end{pmatrix} \tag{B.8}$$

$$-\frac{\partial \vec{f}}{\partial \vec{u}}\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T = \begin{pmatrix} -1 & -b(1-x^2) \\ b(1-x^2) & -(1+2bxy)^2 - b^2(1-x^2)^2 \end{pmatrix} \tag{B.9}$$

$$-\frac{1}{\alpha^2}\vec{f}\vec{f}^T = -\frac{1}{\alpha^2}\begin{pmatrix} f_1^2 & f_1 f_2 \\ f_2 f_1 & f_2^2 \end{pmatrix} \tag{B.10}$$

Thus:

$$\mathbf{B} = \begin{pmatrix} -1-\frac{1}{\alpha^2}f_1^2 & -2b\left(\frac{x_i-x_{i-1}}{\Delta t}y_i + \frac{y_i-y_{i-1}}{\Delta t}x_i\right) - b(1-x^2) - \frac{1}{\alpha^2}f_1 f_2 \\ b(1-x^2) - \frac{1}{\alpha^2}f_2 f_1 & -2bx_i\left(\frac{x_i-x_{i-1}}{\Delta t}\right) - (1+2bxy)^2 - b^2(1-x^2)^2 - \frac{1}{\alpha^2}f_2^2 \end{pmatrix} \tag{B.11}$$

and, identical to the previous section:

$$\vec{C} = \left(0\ (1-x^2)y\right)^T \tag{B.12}$$

## B.3 Sensitivity Derivative

The Sensitivity Derivative is given by differentiating Eq. 3.3 w.r.t. $b$. According to Eq. A.39, for $p = 8$:

$$\frac{\delta F}{\delta b} = \frac{1}{8}\left(\frac{1}{T}\int_0^T y^8\,dt\right)^{-7/8}\left[\overline{\langle\frac{\partial J}{\partial \vec{u}}, v\rangle} + \overline{\eta J} - \overline{\eta}\,\overline{J} + \frac{\partial \overline{J}}{\partial b}\right] \tag{B.13}$$

where $J = y^8$ and $\overline{J} = \frac{1}{T}\int_0^T y^8\,dt$. Also,

$$\frac{\partial J}{\partial \vec{u}} = \left[\frac{\partial y^8}{\partial x}\frac{\partial y^8}{\partial y}\right] = \left[08y^7\right] \tag{B.14}$$

$$\langle\frac{\partial J}{\partial \vec{u}}, v\rangle = \left[08y^7\right]\left[v_x v_y\right]^T = 8y^7 v_y \tag{B.15}$$

$$\overline{\langle\frac{\partial J}{\partial \vec{u}}, v\rangle} = \frac{1}{T}\int_0^T 8y^7 v_y\,dt \tag{B.16}$$

$$\frac{\partial \overline{J}}{\partial b} = 0 \tag{B.17}$$

# Appendix C

# Derivations in the Rossler case

In the Rossler system, the vector of unknowns is: $\vec{u}(t) = [x(t)y(t)z(t)]^T$. The right-hand-side of Eq. 4.1 is:

$$\vec{f} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} = \begin{pmatrix} -y - z \\ x + ay \\ b + z(x - c) \end{pmatrix} \tag{C.1}$$

The Jacobian of Eq. C.1 is:

$$\frac{\partial \vec{f}}{\partial \vec{u}} = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} \end{pmatrix} = \begin{pmatrix} 0 & -1 & -1 \\ 1 & a & 0 \\ z & 0 & x - c \end{pmatrix} \tag{C.2}$$

## C.1 Coefficients of the second Order ODE (LSS)

By substituting the previous section's expressions in Eq. A.18, the formula of the coefficient $\mathbf{A}$ is evaluated:

$$\mathbf{A} = \left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T - \frac{\partial \vec{f}}{\partial \vec{u}} = \begin{pmatrix} 0 & 2 & z + 1 \\ -2 & 0 & 0 \\ -z - 1 & 0 & 0 \end{pmatrix} \tag{C.3}$$

For $\mathbf{B}$, it can be shown that:

$$\mathbf{B} = \begin{pmatrix} -2 - \frac{1}{\alpha^2} f_1^2 & a - \frac{1}{\alpha^2} f_1 f_2 & b + (z+1) * (x-c) - \frac{1}{\alpha^2} f_1 f_3 \\ -a - \frac{1}{\alpha^2} f_2 f_1 & -1 - a^2 - \frac{1}{\alpha^2} f_2^2 & -z - \frac{1}{\alpha^2} f_2 f_3 \\ x - c - \frac{1}{\alpha^2} f_3 f_1 & -z - \frac{1}{\alpha^2} f_3 f_2 & -y - z - z^2 - (x-c)^2 - \frac{1}{\alpha^2} f_3^2 \end{pmatrix} \quad \text{(C.4)}$$

Also:

$$\vec{C} = (0\, y\, 0)^T \quad \text{(C.5)}$$

where $\frac{dx}{dt}$, $\frac{dy}{dt}$ and $\frac{dz}{dt}$ were substituted using the analytical expression of Eq. 4.1.

## C.2 Coefficients of the second Order ODE (DCLSS)

The coefficients $\mathbf{A_1}$ and $\mathbf{A_2}$ of Eq. A.51 are:

$$\mathbf{A_1} = \left( \frac{\partial \vec{f}}{\partial \vec{u}} \right)^T = \begin{pmatrix} 0 & 1 & z \\ -1 & a & 0 \\ -1 & 0 & x - c \end{pmatrix} \quad \text{(C.6)}$$

$$\mathbf{A_2} = -\frac{\partial \vec{f}}{\partial \vec{u}} = - \begin{pmatrix} 0 & 1 & 1 \\ -1 & -a & 0 \\ -z & 0 & c - x \end{pmatrix} \quad \text{(C.7)}$$

It is true that:

$$\frac{\overleftarrow{d}}{dt} \left( \frac{\partial \vec{f}}{\partial \vec{u}} \right)^T = \begin{pmatrix} 0 & 0 & \frac{dz}{dt} \\ 0 & 0 & 0 \\ 0 & 0 & \frac{dx}{dt} \end{pmatrix} = \begin{pmatrix} 0 & 0 & \frac{z_i - z_{i-1}}{dt} \\ 0 & 0 & 0 \\ 0 & 0 & \frac{x_i - x_{i-1}}{dt} \end{pmatrix} \quad \text{(C.8)}$$

$$-\frac{\partial \vec{f}}{\partial \vec{u}} \left( \frac{\partial \vec{f}}{\partial \vec{u}} \right)^T = \begin{pmatrix} 2 & -1 & c - x \\ a & 1 + a^2 & z \\ c - x & z & z^2 + (c - x)^2 \end{pmatrix} \quad \text{(C.9)}$$

$$-\frac{1}{a^2} \vec{f} \vec{f}^T = -\frac{1}{a^2} \begin{pmatrix} f_1^2 & f_1 f_2 & f_3 \\ f_2 f_1 & f_2^2 & f_2 f_3 \\ f_3 f_1 & f_3 f_2 & f_3^2 \end{pmatrix} \quad \text{(C.10)}$$

Thus:

$$\mathbf{B} = \begin{pmatrix} -2 - \frac{1}{\alpha^2} f_1^2 & a - \frac{1}{\alpha^2} f_1 f_2 & \frac{z_i - z_{i-1}}{dt} + x - c - \frac{1}{\alpha^2} f_1 f_3 \\ -a - \frac{1}{\alpha^2} f_2 f_1 & -1 - a^2 - \frac{1}{\alpha^2} f_2^2 & -z - \frac{1}{\alpha^2} f_2 f_3 \\ x - c - \frac{1}{\alpha^2} f_1 f_3 & -z - \frac{1}{\alpha^2} f_3 f_2 & \frac{x_i - x_{i-1}}{dt} - z^2 - (c - x)^2 - \frac{1}{\alpha^2} f_3^2 \end{pmatrix} \tag{C.11}$$

$$\vec{C} = (0\ y\ 0)^T \tag{C.12}$$

## C.3 Sensitivity Derivative

The Sensitivity Derivative is given by differentiating Eq. 4.2 w.r.t. $a$. The objective function is identical to the one from the Lorenz '63 case, and so is the formula for the SD:

$$\frac{\delta F}{\delta a} = \frac{1}{T} \int_0^T (v_z + \eta z)\ dt - \frac{1}{T^2} \int_0^T \eta\, dt \int_0^T z\, dt \tag{C.13}$$

# Appendix D

# Formulas Used in Data Assimilation Derivations

## D.1 Sample Variance

Supposing a set of $N$ samples $x_i$ with mean value $\overline{x} = \sum_{i=1}^{N} x_i$, the variance $\sigma^2$ can be calculated from:

$$\sigma^2 = E\left[\left(x - E\left[x\right]\right)^2\right] \approx \overline{\left(x - \overline{x}\right)^2} = \frac{1}{N-1}\sum_{i=1}^{N}\left(x_i - \overline{x}\right)^2 \tag{D.1}$$

It should be noted that in the denominator of Eq. D.1 $N-1$ is used instead of $N$. This is called Bessel's correction, and it aims to correct a bias that arises when estimating the variance of a population from a sample [12].

## D.2 Covariance Matrix

The auto-covariance matrix for a sample $\vec{x}$ can be calculated as:

$$\mathbf{C} = E\left[\left(\vec{x} - E\left[\vec{x}\right]\right)\left(\vec{x} - E\left[\vec{x}\right]\right)^T\right] \approx \overline{\left(\vec{x} - \overline{\vec{x}}\right)\left(\vec{x} - \overline{\vec{x}}\right)^T} = \frac{1}{N-1}\sum_{i=1}^{N}\left(\vec{x} - \overline{\vec{x}}\right)\left(\vec{x} - \overline{\vec{x}}\right)^T \tag{D.2}$$

The auto-covariance matrix contains the covariances between each pair of scalar elements of $\vec{x}$ [12]. When two statistical quantities, vector or scalar, are uncorrelated, their covariance matrix is zero projected on a proper dimension.

## D.3 Trace of a Matrix

The trace of a $(n \times n)$ square matrix $\mathbf{A}$, denoted as $tr\,(\mathbf{A})$ is the sum of the elements of its main diagonal:

$$tr\,(\mathbf{A}) = \sum_{i=1}^{n} a_{ii} \tag{D.3}$$

where $a_{ii}$ denotes the entry on the $i_{th}$ row and $i_{th}$ column of $\mathbf{A}$. Some of the basic properties of the trace of a matrix are shown below [16]. Assuming $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{X}$ are matrices of dimensions such that the following expressions are defined, and $a$ and $b$ are scalars:

- Linearity: $tr\,(a\mathbf{A} + b\mathbf{A}) = atr(\mathbf{A}) + btr(\mathbf{B})$
- Immune to transposition: $tr\left(\mathbf{A}^T\right) = tr\,(\mathbf{A})$
- Immune to rotation: $tr\,(\mathbf{ABX}) = tr\,(\mathbf{BXA}) = tr\,(\mathbf{XAB})$

Also, the following identities hold for derivatives of matrices involving traces:

- $\frac{\partial}{\partial \mathbf{X}}\left[\mathbf{X}\right] = \mathbf{I}$
- $\frac{\partial}{\partial \mathbf{X}}\left[\mathbf{XA}\right] = \mathbf{A}^T$
- $\frac{\partial}{\partial \mathbf{X}}\left[\mathbf{AX}^T\right] = \mathbf{A}$
- $\frac{\partial}{\partial \mathbf{X}}\left[\mathbf{XAX}^T\right] = \mathbf{X}\left(\mathbf{A}^T + \mathbf{A}\right)$

## D.4 Derivation of the Extended Kalmal Filter (EKF)

Having derived the assimilated state as a function of $\vec{u}^{\,m}$, $\vec{v}$ and $\mathbf{K}$, the value of the latter is not yet known, the second condition, which states that the assimilated state error $\vec{e}^{\,a}$ needs to have the minimum variance among all other estimates of the true state, has to be utilized. This suggests that the sum of the diagonal elements of the error auto-covariance matrix $\mathbf{C^a}$, i.e its trace $tr\,(\mathbf{C^a})$ which consists of the squared sum of the variances of each element of $\vec{e}^{\,a}$, needs to be minimized. In order

to derive $\mathbf{C^a}$, $\vec{e}^{\,a}$ has to be derived first. For the present state, at time $k$:

$$\begin{aligned}
\vec{e}_k^{\,a} &= \vec{u}_k^{\,t} - \vec{u}_k^{\,a} \\
&= M\left(\vec{u}_{k-1}^{\,t}\right) + \vec{e}_{k-1}^{\,m} - \vec{u}_k^{\,m} - \mathbf{K}_k\left(\vec{v}_k - H\left(\vec{u}_k^{\,m}\right)\right) \\
&\overset{5.1}{=} M\left(\vec{u}_{k-1}^{\,t}\right) - M\left(\vec{u}_{k-1}^{\,a}\right) + \vec{e}_{k-1}^{\,m} - \mathbf{K}_k\left(H\left(\vec{u}_k^{\,t}\right) + \vec{e}_k^{\,o} - H\left(\vec{u}_k^{\,m}\right)\right) \\
&\overset{D.8}{=} \nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a} + \vec{e}_{k-1}^{\,m} - \mathbf{K}_k\left(H\left(\vec{u}_k^{\,t}\right) + \vec{e}_k^{\,o} - H\left(\vec{u}_k^{\,m}\right)\right) \\
&\overset{D.11}{=} \nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a} + \vec{e}_{k-1}^{\,m} - \mathbf{K}_k\left(\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\,\vec{e}_k^{\,m} + \vec{e}_k^{\,o}\right) \\
&= \nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a} + \vec{e}_{k-1}^{\,m} - \mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\,\vec{e}_k^{\,m} - \mathbf{K}_k\vec{e}_k^{\,o} \\
&\overset{D.10}{=} \nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a} + \vec{e}_{k-1}^{\,m} - \mathbf{K}_k\left(\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a} + \vec{e}_{k-1}^{\,m}\right) + \vec{e}_k^{\,o}\right)
\end{aligned} \tag{D.4}$$

Solving Eq. D.4 for $\vec{e}_k^{\,a}$ gives:

$$\vec{e}_k^{\,a} = \left(\mathbf{I} - \mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a} + \vec{e}_{k-1}^{\,m}\right) - \mathbf{K}_k\vec{e}_k^{\,o} \tag{D.5}$$

In the derivation of Eq. D.4 the following expressions were used:

$$\vec{u}_k^{\,m} = M\left(\vec{u}_{k-1}^{\,a}\right) \tag{D.6}$$

By subtracting Eq. 5.1 from Eq. 5.3:

$$\begin{aligned}
\vec{u}_k^{\,t} - \vec{u}_k^{\,m} = M\left(\vec{u}_{k-1}^{\,t}\right) - M\left(\vec{u}_{k-1}^{\,a}\right) + \vec{e}_{k-1}^{\,m} \Rightarrow \\
\vec{e}_k^{\,m} = M\left(\vec{u}_{k-1}^{\,t}\right) - M\left(\vec{u}_{k-1}^{\,a}\right) + \vec{e}_{k-1}^{\,m}
\end{aligned} \tag{D.7}$$

Similar to $H(\cdot)$, the non-linear operator $M(\cdot)$ can be linearized around $\vec{u}_{k-1}^{\,a}$ as:

$$\begin{aligned}
&M\left(\vec{u}_{k-1}^{\,t}\right) = M\left(\vec{u}_{k-1}^{\,a}\right) + \nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\left(\vec{u}_{k-1}^{\,t} - \vec{u}_{k-1}^{\,a}\right) + O\left(\left(\vec{u}_{k-1}^{\,t} - \vec{u}_{k-1}^{\,m}\right)^2\right) \Rightarrow \\
&M\left(\vec{u}_{k-1}^{\,t}\right) \approx M\left(\vec{u}_{k-1}^{\,a}\right) + \nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\left(\vec{u}_{k-1}^{\,t} - \vec{u}_{k-1}^{\,a}\right) \iff \\
&M\left(\vec{u}_{k-1}^{\,t}\right) \approx M\left(\vec{u}_{k-1}^{\,a}\right) + \nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a}
\end{aligned} \tag{D.8}$$

where:

$$\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}} = \left.\frac{\partial M}{\partial \vec{u}}\right|_{\vec{u}=\vec{u}_{k-1}^{\,a}} \tag{D.9}$$

Thus, Eq. D.7 becomes:

$$\vec{e}_k^{\,m} = \nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a} + \vec{e}_{k-1}^{\,m} \tag{D.10}$$

Similarly to Eq. D.8:

$$H\left(\vec{u}_k^{\,t}\right) \approx H\left(\vec{u}_k^{\,m}\right) + \nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\, \vec{e}_k^{\,m} \tag{D.11}$$

where:

$$\nabla\mathbf{H}|_{\vec{u}_k^{\,m}} = \left.\frac{\partial H}{\partial \vec{u}}\right|_{\vec{u}=\vec{u}_k^{\,m}} \tag{D.12}$$

Having derived $\vec{e}_k^{\,a}$, $\mathbf{C^a}_k = E\left(\vec{e}_k^{\,a}\vec{e}_k^{\,a\,T}\right)$ can now also be derived, so that the value of $\mathbf{K}_k$ can be computed such that it minimizes $\|\vec{e}_k^{\,a}\|^2 = \sum_i \left(\vec{e}_{k,i}^{\,a}\right)^2 = tr\left(\mathbf{C^a}_k\right)$, as mentioned previously.

$$\begin{aligned}
\vec{e}_k^{\,a}\vec{e}_k^{\,a\,T} &= \\
&= \left[\left(\mathbf{I} - \mathbf{K}_k\, \nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\, \vec{e}_{k-1}^{\,a} + \vec{e}_{k-1}^{\,m}\right) - \mathbf{K}_k\vec{e}_k^{\,o}\right] \\
&\quad \left[\left(\mathbf{I} - \mathbf{K}_k\, \nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\, \vec{e}_{k-1}^{\,a} + \vec{e}_{k-1}^{\,m}\right) - \mathbf{K}_k\vec{e}_k^{\,o}\right]^T \\
&= \left[\left(\mathbf{I} - \mathbf{K}_k\, \nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\, \vec{e}_{k-1}^{\,a} + \vec{e}_{k-1}^{\,m}\right) - \mathbf{K}_k\vec{e}_k^{\,o}\right] \\
&\quad \left[\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\, \vec{e}_{k-1}^{\,a} + \vec{e}_{k-1}^{\,m}\right)^T\left(\mathbf{I} - \mathbf{K}_k\, \nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)^T - \vec{e}_k^{\,o\,T}\mathbf{K}_k^T\right] \\
&= \mathbf{T1} + \mathbf{T2} + \mathbf{T3} + \mathbf{T4}
\end{aligned} \tag{D.13}$$

where:

$$\begin{aligned}
\mathbf{T1} &= \left(\mathbf{I} - \mathbf{K}_k\, \nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\, \vec{e}_{k-1}^{\,a} + \vec{e}_{k-1}^{\,m}\right) \\
&\quad \left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\, \vec{e}_{k-1}^{\,a} + \vec{e}_{k-1}^{\,m}\right)^T\left(\mathbf{I} - \mathbf{K}_k\, \nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)^T \\
\mathbf{T2} &= -\mathbf{K}_k\vec{e}_k^{\,o}\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\, \vec{e}_{k-1}^{\,a} + \vec{e}_{k-1}^{\,m}\right)^T\left(\mathbf{I} - \mathbf{K}_k\, \nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)^T \\
\mathbf{T3} &= -\left(\mathbf{I} - \mathbf{K}_k\, \nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\, \vec{e}_{k-1}^{\,a} + \vec{e}_{k-1}^{\,m}\right)\vec{e}_k^{\,o\,T}\mathbf{K}_k^T \\
\mathbf{T4} &= \mathbf{K}_k\vec{e}_k^{\,o}\vec{e}_k^{\,o\,T}\mathbf{K}_k^T
\end{aligned} \tag{D.14}$$

It follows that:

$$\mathbf{C^a}_k = E\left(\vec{e}_k^{\,a}\vec{e}_k^{\,a\,T}\right) = E\left(\mathbf{T1}\right) + E\left(\mathbf{T2}\right) + E\left(\mathbf{T3}\right) + E\left(\mathbf{T4}\right) \tag{D.15}$$

The expectation of **T1** can be written as:

$$
\begin{aligned}
&E\left(\mathbf{T1}\right) = \\
&= E\left(\left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a}+\vec{e}_{k-1}^{\,m}\right)\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a}+\vec{e}_{k-1}^{\,m}\right)^T\left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)^T\right)\\
&= E\left(\left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a}+\vec{e}_{k-1}^{\,m}\right)\left(\vec{e}_{k-1}^{\,a\,T}\,\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}^T+\vec{e}_{k-1}^{\,m\,T}\right)\left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)^T\right)\\
&= \left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)E\left(\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a}+\vec{e}_{k-1}^{\,m}\right)\left(\vec{e}_{k-1}^{\,a\,T}\,\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}^T+\vec{e}_{k-1}^{\,m\,T}\right)\right)\left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)^T\\
&= \left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)E\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a}\vec{e}_{k-1}^{\,a\,T}\,\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}^T+\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a}\vec{e}_{k-1}^{\,m\,T}+\right.\\
&\quad\left.\vec{e}_{k-1}^{\,m}\vec{e}_{k-1}^{\,a\,T}\,\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}^T+\vec{e}_{k-1}^{\,m}\vec{e}_{k-1}^{\,m\,T}\right)\left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)^T\\
&= \left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,E\left(\vec{e}_{k-1}^{\,a}\vec{e}_{k-1}^{\,a\,T}\right)\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}^T+\mathbf{0}+\mathbf{0}+\mathbf{C}_{\mathbf{k-1}}^{\mathbf{m}}\right)\left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)^T\\
&= \left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\mathbf{C}_{\mathbf{k-1}}^{\mathbf{a}}\,\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}^T+\mathbf{C}_{\mathbf{k-1}}^{\mathbf{m}}\right)\left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)^T
\end{aligned}
\tag{D.16}
$$

Keep in mind that, in Eq. D.16, the uncorrelated terms, namely $\vec{e}_{k-1}^{\,a}\vec{e}_{k-1}^{\,m\,T}$ and $\vec{e}_{k-1}^{\,m}\vec{e}_{k-1}^{\,a\,T}$, have zero covariance thus they cancel out. This is the case, as the errors are not directly dependent to one another. The expectation of the second term can be written as:

$$
\begin{aligned}
E\left(\mathbf{T2}\right) = &\\
&= E\left(-\mathbf{K}_k\vec{e}_k^{\,o}\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a}+\vec{e}_{k-1}^{\,m}\right)^T\left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)^T\right)\\
&= E\left(-\mathbf{K}_k\vec{e}_k^{\,o}\left(\vec{e}_{k-1}^{\,a\,T}\,\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}^T+\vec{e}_{k-1}^{\,m\,T}\right)\right)\left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)^T\\
&= E\left(-\mathbf{K}_k\vec{e}_k^{\,o}\vec{e}_{k-1}^{\,a\,T}\,\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}^T-\mathbf{K}_k\vec{e}_k^{\,o}\vec{e}_{k-1}^{\,a\,T}\vec{e}_{k-1}^{\,m\,T}\right)\left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)^T\\
&= \left(-\mathbf{K}_k E\left(\vec{e}_k^{\,o}\vec{e}_{k-1}^{\,a\,T}\right)\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}^T-\mathbf{K}_k E\left(\vec{e}_k^{\,o}\vec{e}_{k-1}^{\,a\,T}\right)\vec{e}_{k-1}^{\,m\,T}\right)\left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)^T\\
&= \left(-\mathbf{K}_k\cdot\mathbf{0}\,\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}^T-\mathbf{K}_k\cdot\mathbf{0}\vec{e}_{k-1}^{\,m\,T}\right)\left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)^T\\
&= \mathbf{0}\cdot\left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)^T\\
&= \mathbf{0}
\end{aligned}
\tag{D.17}
$$

The expectation of the third term can be written as:

$$
\begin{aligned}
E\left(\mathbf{T3}\right) = & \\
&= E\left(-\left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a}+\vec{e}_{k-1}^{\,m}\right)\vec{e}_k^{\,o\,T}\mathbf{K}_k^T\right)\\
&= -\left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)E\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a}\vec{e}_k^{\,o\,T}\mathbf{K}_k^T+\vec{e}_{k-1}^{\,m}\vec{e}_k^{\,o\,T}\mathbf{K}_k^T\right)\\
&= -\left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}E\left(\vec{e}_{k-1}^{\,a}\vec{e}_k^{\,o\,T}\right)\mathbf{K}_k^T+E\left(\vec{e}_{k-1}^{\,m}\vec{e}_k^{\,o\,T}\right)\mathbf{K}_k^T\right)\\
&= -\left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\mathbf{0}\mathbf{K}_k^T+\mathbf{0}\mathbf{K}_k^T\right)\\
&= -\left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)\mathbf{0}\\
&= \mathbf{0}
\end{aligned}
\tag{D.18}
$$

Finally, the expectation of the third term becomes:

$$
\begin{aligned}
E\left(\mathbf{T4}\right) = & \\
&= E\left(\mathbf{K}_k\vec{e}_k^{\,o}\vec{e}_k^{\,o\,T}\mathbf{K}_k^T\right)\\
&= \mathbf{K}_k E\left(\vec{e}_k^{\,o}\vec{e}_k^{\,o\,T}\right)\mathbf{K}_k^T\\
&= \mathbf{K}_k\mathbf{C}_\mathbf{k}^\mathbf{o}\mathbf{K}_k^T
\end{aligned}
\tag{D.19}
$$

Combining the expectations of the four terms yields $\mathbf{C}_\mathbf{k}^\mathbf{a}$:

$$
\mathbf{C}_\mathbf{k}^\mathbf{a} = \left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\mathbf{C}_\mathbf{k-1}^\mathbf{a}\,\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}^T+\mathbf{C}_\mathbf{k-1}^\mathbf{m}\right)\left(\mathbf{I}-\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)^T+\mathbf{K}_k\mathbf{C}_\mathbf{k}^\mathbf{o}\mathbf{K}_k^T \tag{D.20}
$$

In order to further simplify Eq. D.20, it can be proven that:

$$
\mathbf{C}_\mathbf{k}^\mathbf{m} = \nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\mathbf{C}_\mathbf{k-1}^\mathbf{a}\,\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}^T+\mathbf{C}_\mathbf{k-1}^\mathbf{m} \tag{D.21}
$$

$$
\begin{aligned}
\mathbf{C_k^m} &= E\left(\vec{e}_k^{\,m}\vec{e}_k^{\,m\,T}\right) = \\
&= E\left(\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a} + \vec{e}_{k-1}^{\,m}\right)\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a} + \vec{e}_{k-1}^{\,m}\right)^T\right) \\
&= E\left(\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a} + \vec{e}_{k-1}^{\,m}\right)\left(\vec{e}_{k-1}^{\,a\,T}\,\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}^T + \vec{e}_{k-1}^{\,m\,T}\right)\right) \\
&= E\left(\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a}\vec{e}_{k-1}^{\,a\,T}\,\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}^T + \nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\vec{e}_{k-1}^{\,a}\vec{e}_{k-1}^{\,m\,T} + \vec{e}_{k-1}^{\,m}\vec{e}_{k-1}^{\,a\,T}\,\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}^T + \vec{e}_{k-1}^{\,m}\vec{e}_{k-1}^{\,m\,T}\right) \\
&= \nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,E\left(\vec{e}_{k-1}^{\,a}\vec{e}_{k-1}^{\,a\,T}\right)\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}^T + \nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,E\left(\vec{e}_{k-1}^{\,a}\vec{e}_{k-1}^{\,m\,T}\right) + \\
&\quad E\left(\vec{e}_{k-1}^{\,m}\vec{e}_{k-1}^{\,a\,T}\right)\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}^T + E\left(\vec{e}_{k-1}^{\,m}\vec{e}_{k-1}^{\,m\,T}\right) \\
&= \nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\mathbf{C_{k-1}^a}\,\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}^T + \nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\cdot\mathbf{0} + \mathbf{0}\cdot\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}^T + \mathbf{C_{k-1}^m} \\
&= \nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}\,\mathbf{C_{k-1}^a}\,\nabla\mathbf{M}|_{\vec{u}_{k-1}^{\,a}}^T + \mathbf{C_{k-1}^m}
\end{aligned}
\tag{D.22}
$$

Thus, Eq. D.20 can be written as:

$$
\mathbf{C_k^a} = \left(\mathbf{I} - \mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)\mathbf{C_k^m}\left(\mathbf{I} - \mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)^T + \mathbf{K}_k\mathbf{C_k^o}\mathbf{K}_k^T \tag{D.23}
$$

In order to derive $tr\left(\mathbf{C_k^a}\right)$, $\mathbf{C_k^a}$ is expanded:

$$
\begin{aligned}
\mathbf{C_k^a} &= \left(\mathbf{C_k^m} - \mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\,\mathbf{C_k^m}\right)\left(\mathbf{I} - \mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\right)^T + \mathbf{K}_k\mathbf{C_k^o}\mathbf{K}_k^T \\
&= \left(\mathbf{C_k^m} - \mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\,\mathbf{C_k^m}\right)\left(\mathbf{I} - \nabla\mathbf{H}|_{\vec{u}_k^{\,m}}^T\,\mathbf{K}_k^T\right) + \mathbf{K}_k\mathbf{C_k^o}\mathbf{K}_k^T \\
&= \mathbf{C_k^m} - \mathbf{C_k^m}\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}^T\,\mathbf{K}_k^T - \mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\,\mathbf{C_k^m} + \mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\,\mathbf{C_k^m} + \\
&\quad \mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\,\mathbf{C_k^m}\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}^T\,\mathbf{K}_k^T + \mathbf{K}_k\mathbf{C_k^o}\mathbf{K}_k^T
\end{aligned}
\tag{D.24}
$$

Now that the assimilated state auto-covariance matrix $\mathbf{C_k^a}$ is known, its trace $tr\left(\mathbf{C_k^a}\right)$, i.e the squared sums of the assimilated state error vector components, is also known. In order to determine the value of $\mathbf{K}_k$ that minimizes the trace, $\frac{\partial tr\left(\mathbf{C_k^a}\right)}{\partial\mathbf{K}_k}$ is set to zero.

$$
\begin{aligned}
\frac{\partial tr\left(\mathbf{C_k^a}\right)}{\partial\mathbf{K}_k} = &-\frac{\partial}{\partial\mathbf{K}_k}tr\left(\left(\mathbf{C_k^m}\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}^T\right)\mathbf{K}_k^T\right) \\
&-\frac{\partial}{\partial\mathbf{K}_k}tr\left(\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\,\mathbf{C_k^m}\right) \\
&+\frac{\partial}{\partial\mathbf{K}_k}tr\left(\mathbf{K}_k\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}\,\mathbf{C_k^m}\,\nabla\mathbf{H}|_{\vec{u}_k^{\,m}}^T\,\mathbf{K}_k^T\right) \\
&+\frac{\partial}{\partial\mathbf{K}_k}tr\left(\mathbf{K}_k\mathbf{C_k^o}\mathbf{K}_k^T\right)
\end{aligned}
\tag{D.25}
$$

The first term is equivalent to:

$$-\frac{\partial}{\partial \mathbf{K}_k} tr\left(\left(\mathbf{C_k^m}\ \nabla \mathbf{H}|_{\vec{u}_k^m}^T\right)\mathbf{K}_k^T\right) \overset{D.3}{=} -\mathbf{C_k^m}\ \nabla \mathbf{H}|_{\vec{u}_k^m}^T \tag{D.26}$$

The second term is equivalent to:

$$-\frac{\partial}{\partial \mathbf{K}_k} tr\left(\mathbf{K}_k\ \nabla \mathbf{H}|_{\vec{u}_k^m}\ \mathbf{C_k^m}\right) \overset{D.3}{=} -\left(\nabla \mathbf{H}|_{\vec{u}_k^m}\ \mathbf{C_k^m}\right)^T \tag{D.27}$$

The third term is equivalent to:

$$\frac{\partial}{\partial \mathbf{K}_k} tr\left(\mathbf{K}_k\ \nabla \mathbf{H}|_{\vec{u}_k^m}\ \mathbf{C_k^m}\ \nabla \mathbf{H}|_{\vec{u}_k^m}^T\ \mathbf{K}_k^T\right) \overset{D.3}{=} \mathbf{K}_k\left[\left(\nabla \mathbf{H}|_{\vec{u}_k^m}\ \mathbf{C_k^m}\ \nabla \mathbf{H}|_{\vec{u}_k^m}^T\right) + \left(\nabla \mathbf{H}|_{\vec{u}_k^m}\ \mathbf{C_k^m}\ \nabla \mathbf{H}|_{\vec{u}_k^m}^T\right)^T\right] \tag{D.28}$$

Since $\nabla \mathbf{H}|_{\vec{u}_k^m}$ and $\mathbf{C_k^m}$ are both symmetric matrices by definition, meaning that $\nabla \mathbf{H}|_{\vec{u}_k^m} = \nabla \mathbf{H}|_{\vec{u}_k^m}^T$ and $\mathbf{C_k^m} = \mathbf{C_k^{m}}^T$, it will be shown that $\left(\nabla \mathbf{H}|_{\vec{u}_k^m}\ \mathbf{C_k^m}\ \nabla \mathbf{H}|_{\vec{u}_k^m}^T\right)$ is also a symmetric matrix, and thus Eq. D.28 can further be simplified to:

$$\begin{aligned}&\left(\nabla \mathbf{H}|_{\vec{u}_k^m}\ \mathbf{C_k^m}\ \nabla \mathbf{H}|_{\vec{u}_k^m}^T\right)^T = \left(\nabla \mathbf{H}|_{\vec{u}_k^m}\left(\mathbf{C_k^m}\ \nabla \mathbf{H}|_{\vec{u}_k^m}^T\right)\right)^T = \\ &\left(\mathbf{C_k^m}\ \nabla \mathbf{H}|_{\vec{u}_k^m}^T\right)^T \nabla \mathbf{H}|_{\vec{u}_k^m}^T = \nabla \mathbf{H}|_{\vec{u}_k^m}\ \mathbf{C_k^{m}}^T\ \nabla \mathbf{H}|_{\vec{u}_k^m}^T = \nabla \mathbf{H}|_{\vec{u}_k^m}\ \mathbf{C_k^m}\ \nabla \mathbf{H}|_{\vec{u}_k^m}^T\end{aligned} \tag{D.29}$$

As a result, Eq. D.28 can be expressed as:

$$\frac{\partial}{\partial \mathbf{K}_k} tr\left(\mathbf{K}_k\ \nabla \mathbf{H}|_{\vec{u}_k^m}\ \mathbf{C_k^m}\ \nabla \mathbf{H}|_{\vec{u}_k^m}^T\ \mathbf{K}_k^T\right) = 2\mathbf{K}_k\left(\nabla \mathbf{H}|_{\vec{u}_k^m}\ \mathbf{C_k^m}\ \nabla \mathbf{H}|_{\vec{u}_k^m}^T\right) \tag{D.30}$$

Similarly, since $\mathbf{C_k^o}$ is also symmetric by definition, the fourth term is equivalent to:

$$\frac{\partial}{\partial \mathbf{K}_k} tr\left(\mathbf{K}_k \mathbf{C_k^o} \mathbf{K}_k^T\right) \overset{D.3}{=} 2\mathbf{K}_k \mathbf{C_k^o} \tag{D.31}$$

Overall, by setting $\frac{\partial tr\left(\mathbf{C_k^a}\right)}{\partial \mathbf{K}_k} = \mathbf{0}$:

$$\begin{aligned}&2\mathbf{K}_k\left[\nabla \mathbf{H}|_{\vec{u}_k^m}\ \mathbf{C_k^m}\ \nabla \mathbf{H}|_{\vec{u}_k^m}^T + \mathbf{C_k^o}\right] = \left(\nabla \mathbf{H}|_{\vec{u}_k^m}\ \mathbf{C_k^m}\right)^T + \mathbf{C_k^m}\ \nabla \mathbf{H}|_{\vec{u}_k^m}^T \Rightarrow \\ &2\mathbf{K}_k\left[\nabla \mathbf{H}|_{\vec{u}_k^m}\ \mathbf{C_k^m}\ \nabla \mathbf{H}|_{\vec{u}_k^m}^T + \mathbf{C_k^o}\right] = \mathbf{C_k^{m}}^T\ \nabla \mathbf{H}|_{\vec{u}_k^m}^T + + \mathbf{C_k^m}\ \nabla \mathbf{H}|_{\vec{u}_k^m}^T \overset{\text{symmetry}}{\Rightarrow} \\ &2\mathbf{K}_k\left[\nabla \mathbf{H}|_{\vec{u}_k^m}\ \mathbf{C_k^m}\ \nabla \mathbf{H}|_{\vec{u}_k^m}^T + \mathbf{C_k^o}\right] = 2\mathbf{C_k^m}\ \nabla \mathbf{H}|_{\vec{u}_k^m}^T\end{aligned} \tag{D.32}$$

And, finally, the Extended Kalman Filter (EKF) $\mathbf{K}$ corresponding to time-step $k$, where observations are available and data assimilation is performed, is given by:

$$\mathbf{K} = \mathbf{C^m}\, \nabla \mathbf{H}|_{\vec{u}\,m}^{T} \left[ \nabla \mathbf{H}|_{\vec{u}\,m}\, \mathbf{C^m}\, \nabla \mathbf{H}|_{\vec{u}\,m}^{T} + \mathbf{C^o} \right]^{-1} \tag{D.33}$$

All quantities of Eq. 5.20 refer to time-step $k$, so the corresponding index is omitted. Therefore, $\mathbf{K}$ depends on $H(\cdot)$, $\mathbf{C^o}$ and $\mathbf{C^m}$. The auto-covariance matrices $\mathbf{C^o}$ and $\mathbf{C^m}$ are of known value, given by Eq. 5.7 and 5.8 respectively. Also, the assimilated state error auto-covariance matrix $\mathbf{C^a}$ can be derived:

$$\begin{aligned}
\mathbf{C^a} &= (\mathbf{I} - \mathbf{K}\, \nabla \mathbf{H}|_{\vec{u}\,m})\, \mathbf{C^m}\, (\mathbf{I} - \mathbf{K}\, \nabla \mathbf{H}|_{\vec{u}\,m})^T + \mathbf{K}\mathbf{C^o}\mathbf{K}^T \\
&= (\mathbf{I} - \mathbf{K}\, \nabla \mathbf{H}|_{\vec{u}\,m})\, \mathbf{C^m} \left( \mathbf{I} - \nabla \mathbf{H}|_{\vec{u}\,m}^{T}\, \mathbf{K}^T \right) + \mathbf{K}\mathbf{C^o}\mathbf{K}^T \\
&= (\mathbf{I} - \mathbf{K}\, \nabla \mathbf{H}|_{\vec{u}\,m})\, \mathbf{C^m} - (\mathbf{I} - \mathbf{K}\, \nabla \mathbf{H}|_{\vec{u}\,m})\, \mathbf{C^m}\, \nabla \mathbf{H}|_{\vec{u}\,m}^{T}\, \mathbf{K}^T + \mathbf{K}\mathbf{C^o}\mathbf{K}^T \\
&= (\mathbf{I} - \mathbf{K}\, \nabla \mathbf{H}|_{\vec{u}\,m})\, \mathbf{C^m} - \left[ (\mathbf{I} - \mathbf{K}\, \nabla \mathbf{H}|_{\vec{u}\,m})\, \mathbf{C^m}\, \nabla \mathbf{H}|_{\vec{u}\,m}^{T} - \mathbf{K}\mathbf{C^o} \right] \mathbf{K}^T \\
&= (\mathbf{I} - \mathbf{K}\, \nabla \mathbf{H}|_{\vec{u}\,m})\, \mathbf{C^m} - \left[ \mathbf{C^m}\, \nabla \mathbf{H}|_{\vec{u}\,m}^{T} - \mathbf{K} \left( \nabla \mathbf{H}|_{\vec{u}\,m}\, \mathbf{C^m}\, \nabla \mathbf{H}|_{\vec{u}\,m}^{T} + \mathbf{C^o} \right) \right] \mathbf{K}_k^T \\
&\overset{5.20}{=} (\mathbf{I} - \mathbf{K}\, \nabla \mathbf{H}|_{\vec{u}\,m})\, \mathbf{C^m} - \mathbf{0} \\
&= (\mathbf{I} - \mathbf{K}\, \nabla \mathbf{H}|_{\vec{u}\,m})\, \mathbf{C^m}
\end{aligned} \tag{D.34}$$

Thus, the assimilated state error auto-covariance matrix is:

$$\mathbf{C^a} = (\mathbf{I} - \mathbf{K}\, \nabla \mathbf{H}|_{\vec{u}\,m})\, \mathbf{C^m} \tag{D.35}$$

# Bibliography


[1] Aliakbari, M., Mahmoudi, M., Vadasz, P., Arzani, A.: Predicting high-fidelity multiphysics data from low-fidelity fluid flow and transport solvers using physics-informed neural networks (2022)

[2] Ang, E., anf B. NG, G.W.: Physics-informed neural networks for low reynolds number flows over cylinder (2023)

[3] Blonigan, P.J.: New Methods for Sensitivity Analysis of Chaotic Dynamical Systems. Master's thesis, Massachusetts Institute of Technology (2013)

[4] Blonigan, P.J.: Least Squares Shadowing for Sensitivity Analysis of Large Chaotic Systems and Fluid Flows. Ph.D. thesis, Massachusetts Institute of Technology (June 2016)

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

[6] Blonigan, P.J., Gomez, S.A., Wang, Q.: Least squares shadowing for sensitivity analysis of turbulent fluid flows. `https://arxiv.org/abs/1401.4163` (2014)

[7] Blonigan, P.J., Wang, Q.: Least squares shadowing sensitivity analysis of chaotic flow around a two-dimensional airfoil. American Institute of Aeronautics and Astronautics **56**(2) (February 2018)

[8] Cai, S., Wang, Z., Karniadakis, G.: Physics-informed neural networks for heat transfer problems (2021)

[9] Chandramoorthy, N., Wang, Z.N., Wang, Q., Tucker, P.: Toward computing sensitivities of average quantities in turbulent flows. `https://arxiv.org/abs/1902.11112v1` (2018), center for Turbulence Research, Proceedings of the Summer Program 2018, arXiv:1902.11112 [cs.CE]

[10] Chater, M., Ni, A., Blonigan, P.J., Wang, Q.: Least squares shadowing method for sensitivity analysis of differential equations. `https://arxiv.org/abs/1509.02882v2` (2017)

[11] G.A.Terejanu: Extended kalman filter tutorial. `https://homes.cs.washington.edu/~todorov/courses/cseP590/readings/tutorialEKF.pdf`

(2014), center for Turbulence Research, Proceedings of the Summer Program 2018, arXiv:1902.11112 [cs.CE]

[12] G.Evensen: Data Assimilation. Springer, Bergen, Norway

[13] Gomez, S.A.: Parallel Multigrid for Large-Scale Least Squares Sensitivity. Master's thesis, Massachusetts Institute of Technology (June 2013)

[14] Jin, X., Cai, S., Karniadakis, G.: Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations (2021)

[15] Karniadakis, G., Jagtap, A.: Extended physics-informed neural networks (xpinns): A domain decomposition-based solver for nonlinear partial differential equations (2020)

[16] K.B.Petersen, M.S.Petersen: `http://matrixcookbook.com` (2012)

[17] K.C.Giannakoglou: Numerical Analysis for Engineers. Athens, 3rd edn.

[18] K.C.Giannakoglou: Optimization Methods For Engineers. Athens, 4th edn.

[19] Krakos, J.A., Wang, Q., Halland, S.R., Darmofal, D.L.: Sensitivity analysis of limit cycle oscillations. value **231**, 3228–3245 (April 2011)

[20] Law, K., Stuart, A., Zygalakis, K.: Data Assimilation. Springer

[21] Lea, D.J., Allen, M.R., Haine, T.W.: Sensitivity analysis of the climate of a chaotic system. Tellus **32A**, 523–532 (2000)

[22] Lu, L., Meng, X., Mao, Z., Karniadakis, G.: Deepxde: A deep learning library for solving differential equations (2021)

[23] Mao, Z., Jagtap, A., Karniadakis, G.: Physics-informed neural networks for high-speed flows (2020)

[24] McClenny, L., Braga-Neto, U.: Self-adaptive physics-informed neural networks using a soft attention mechanism (2020)

[25] Moseley, B., Markham, A., Nissen-Meyer, T.: Solving the wave equation with physics-informed deep learning (2021)

[26] Ni, A., Wang, Q.: Sensitivity analysis on chaotic dynamical systems by non-intrusive least squares shadowing (nilss). Journal of Computational Physics **347**, 56–77 (2017)

[27] N.J.Kutz: Data-Driven Modeling  Scientific Computation. Oxford University Press, Great Clarendon Street, Oxford, OX2 6DP, United Kingdom

[28] Raissi, M., Perdicaris, P., Karniadakis, G.: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations (2019)

[29] S. Wang, Y. Teng, P.P.: Understanding and mitigating gradient flow pathologies in physics-informed neural networks (2022)

[30] S.E.Simopoulos: Measurements of Technical Quantities. National Techical University of Athens

[31] S.K.Park, Zupanski, M.: Principles of Data Assimilation. Cambridge University Press

[32] Wang, Q.: Forward and adjoint sensitivity computation of chaotic dynamical systems. `https://arxiv.org/abs/1202.5229` (2012)

[33] Wang, Q.: Convergence of the least squares shadowing method for computing derivative of ergodic averages. Society for Industrial and Applied Mathematics (SIAM) **52**, 156–170 (2014)

[34] Wang, Q., Hu, R., Blonigan, P.: Least squares shadowing sensitivity analysis of chaotic limit cycle oscillations. Journal of Computational Physics **267**, 210–224 (2014)

[35] Wei, C., Ooka, R.: Indoor airflow field reconstruction using physics-informed neural network (2023)

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Μηχανολόγων Μηχανικών
Τομέας Ρευστών
Μονάδα Παράλληλης Ρευστοδυναμικής &
Βελτιστοποίησης

# Προγραμματισμός Λογισμικού για την Ανάλυση Χαοτικών Συστημάτων με Μεθόδους Σκίασης Ελαχίστων Τετραγώνων, την Αφομοίωση Δεδομένων και την Επίλυση των Εξισώσεων Ροής Χωρίς Πλέγματα με Νευρωνικά Δίκτυα

Διπλωματική Εργασία

**Γεώργιος Δ. Βάμβουρας**

Επιβλέπων: Κυριάκος Χ. Γιαννάκογλου, Καθηγητής ΕΜΠ

Αθήνα, 2025

# Περιεχόμενα

# Κεφάλαιο 1

# Εισαγωγή

Η διπλωματική εργασία εξετάζει τρεις σύγχρονες υπολογιστικές μεθοδολογίες που αποκτούν αυξανόμενο ενδιαφέρον, στους τομείς της ανάλυσης ευαισθησίας (Sensitivity Analysis) εφαρμόζοντας τον αλγόριθμο Σκίασης Ελαχίστων Τετραγώνων (Least Squares Shadowing, LSS), την Αφομοίωση Δεδομένων (Data Assimilation, DA) για βελτίωση της ακρίβειας πρόβλεψης μοντέλων, και τα Ενημερωμένα από τη Φυσική των Ροών Νευρωνικά Δίκτυα (Physics-Informed Neural Networks, PINNs) για προσομοιώσεις ροής χωρίς πλέγμα.

Οι παραδοσιακές μέθοδοι ανάλυσης ευαισθησίας, όπως οι πεπερασμένες διαφορές και οι συζυγείς μέθοδοι, αποτυγχάνουν σε χαοτικά συστήματα λόγω της εκθετικής απόκλισης των τροχιών με απειροστά διαφορετικές παραμέτρους. Η μέθοδος LSS προσφέρει μια ευσταθή και οικονομική εναλλακτική, αναδιατυπώνοντας την ανάλυση ευαισθησίας ως πρόβλημα βελτιστοποίησης υπό περιορισμούς. Επιπλέον, αναπτύσσεται η Διακριτά Συνεπής LSS (Discretely Consistent LSS, DCLSS), που βελτιώνει την αριθμητική ακρίβεια καθώς εγγυάται συμβατότητα στα σχήματα διακριτοποίησης. Οι δύο παραλλαγές της μεθόδου προγραμματίζονται σε γλώσσα προγραμματισμού C++ και εφαρμόζονται επιτυχώς σε μαθηματικά προβλήματα.

Η εργασία διερευνά επίσης την αφομοίωση δεδομένων για τη βελτίωση των προβλέψεων σε χρονικά μη-μόνιμες προσομοιώσεις. Χρησιμοποιείται το Εκτεταμένο Φίλτρο Kalman (Extended Kalman Filter, EKF) για την ενσωμάτωση μετρήσεων με θόρυβο σε αριθμητικά μοντέλα επίσης με θόρυβο, οδηγώντας σε πιο ακριβή αποτελέσματα από ό,τι οι μετρήσεις ή τα μοντέλα μπορούν ξεχωριστά να προσφέρουν. Προγραμματίζονται σε γλώσσα προγραμματισμού Python εφαρμογές σε μαθηματικά προβλήματα, μέσω των οποίων εξετάζεται η αποτελεσματικότητα της μεθόδου.

Τέλος, εξετάζεται η χρήση των PINNs ως επιλυτών ροής χωρίς τη χρήση πλέγματος ή διακριτοποίησης των εξισώσεων. Τα PINNs ενσωματώνουν τις διαφορικές εξισώσεις και τις συνοριακές συνθήκες που διέπουν ένα πρόβλημα, στη συνάρτηση απωλειών

ενός νευρωνικού δικτύου, χρησιμοποιώντας αυτόματη διαφόριση για τον υπολογισμό παραγώγων. Προγραμματίζονται σε γλώσσα προγραμματισμού Python PINNs δύο ροϊκά προβλήματα: μία ψευδο-1D ασυμπίεστη ροή και μία 2D στρωτή ροή σε αγωγό.

# Κεφάλαιο 2

# Σκίαση Ελαχίστων Τετραγώνων: Εφαρμογή στο Πρόβλημα **Lorenz '63**

Στο κεφάλαιο αυτό παρουσιάζεται η μέθοδος Σκίασης Ελαχίστων Τετραγώνων (Least Squares Shadowing, LSS) για τον υπολογισμό παραγώγων ευαισθησίας της χρονικής μέσης τιμής ποσοτήτων, ειλημμένης σε επαρκή χρονικό ορίζοντα. Για την αποφυγή γενικών μαθηματικών εκφράσεων, γίνεται εφαρμογή πάνω στο πρόβλημα Lorenz 1963, που εκφράζεται ως:

$$\frac{dx}{dt} = \sigma(y - x), \qquad x(0) = x_0 \tag{2.1α′}$$

$$\frac{dy}{dt} = x(\rho - z) - y, \qquad y(0) = y_0 \tag{2.1β′}$$

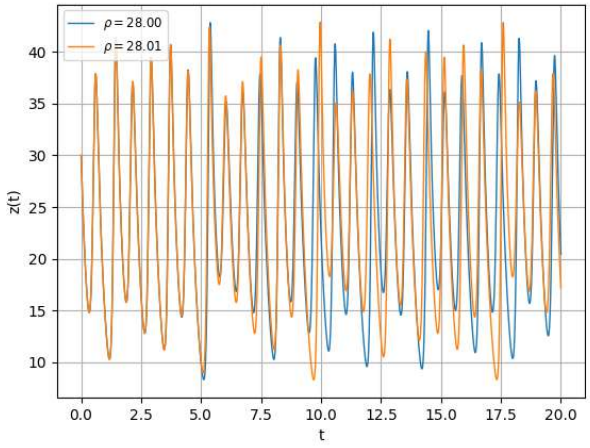
$$\frac{dz}{dt} = xy - \beta z, \qquad z(0) = z_0 \tag{2.1γ′}$$

Η παράμετρος $\rho$ λαμβάνει, σε αυτήν τη διπλωματική, το ρόλο της μεταβλητής σχεδιασμού (design variable) σε ένα πρόβλημα βελτιστοποίησης με αντικειμενική συνάρτηση προς ελαχιστοποίηση την $F$ που είναι η χρονική μέση τιμή του $z(t)$ σε επαρκή χρονικό ορίζοντα:

$$F(\rho) = \frac{1}{T}\int_0^T z(t, \rho)\, dt \tag{2.2}$$

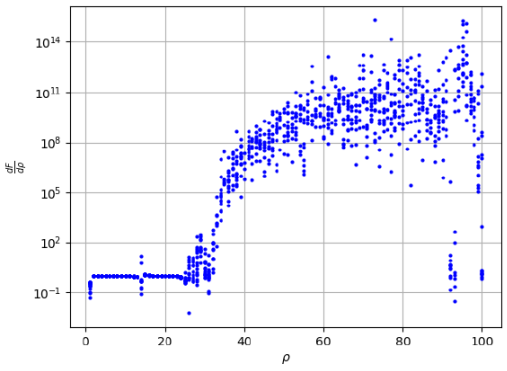
Οι παράμετροι $\sigma$ και $\beta$ είναι σταθερές και ισούνται με 10 και $8/3 \approxeq 2.6667$ αντιστοίχως. Η κατάσταση του συστήματος περιγράφεται από το διάνυσμα $\vec{u}(t, \rho) =$

$[x(t,\rho), y(t,\rho), z(t,\rho)]^T$. Στο Σχ. 2.1 φαίνεται το $z(t)$ για $\rho = 28.00$ και $\rho = 28.01$, με σκοπό να γίνει έμφαση στο ότι οι δύο τροχιές ξεκινούν από το ίδιο σημείο (ίδιες αρχικές συνθήκες) και εξαρχής παραμένουν σχετικά όμοιες αλλά σταδιακά διαφοροποιούνται, ούτως ώστε σε πολύ λίγο χρόνο να είναι τελείως διαφορετικές. Αυτό το φαινόμενο είναι εκδήλωση της χαοτικής φύσης του προβλήματος. Σύμφωνα με τη βιβλιογραφία, η

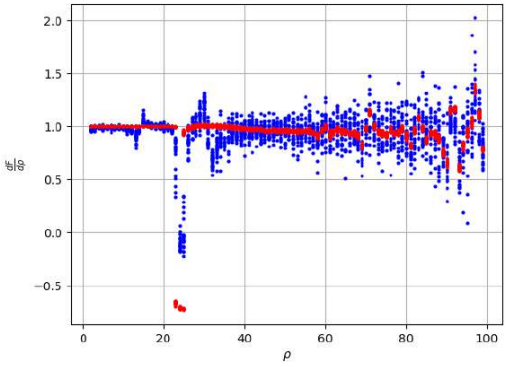



Σχήμα 2.1: *Lorenz '63* Η χρονοσειρά $z(t)$ για $\rho = 28.00$ (σε μπλε) και $\rho = 28.01$ (σε πορτοκαλί).

αντικειμενική συνάρτηση είναι προσεγγιστικά γραμμική, επομένως η παράγωγος ευαισθησίας (SD) $\frac{\delta F}{\delta \rho}$ θα έπρεπε να είναι περίπου ίση με 1.01 [7]. Χρησιμοποιείται η Συνεχής Συζυγής Μέθοδος (Continuous Adjoint Method, CA) για την εύρεση της SD, αλλά αποτυγχάνει να την υπολογίσει εξαιτίας του χαοτικού χαρακτήρα του προβλήματος, και προκύπτουν οι μη-φυσικές τιμές του Σχ. 2.2. Χρησιμοποιούνται επίσης Πεπερασμένες Διαφορές (Finite Differences, FD) με δύο διαφορετικούς χρονικούς ορίζοντες ολοκλήρωσης, $T = 20$ και $T = 2000$ μονάδες χρόνου, οι οποίες δίνουν ικανοποιητική ακρίβεια την αναμενόμενη τιμή της SD, αλλά με ασύμφορο υπολογιστικό κόστος, ακόμα και σε αυτό το απλό μαθηματικό πρόβλημα. Η μέθοδος της Σκίασης Ελαχίστων Τετραγώνων (Least Squares Shadowing, LSS) αποσκοπεί να ξεπεράσει τις δυσκολίες της συζυγών μεθόδων και να παράξει ορθές τιμές παραγώγων ευαισθησίας χρονικά μέσων τιμών ποσοτήτων ως προς κάποια(ες) μεταβλητή(ες) σχεδιασμού, με λιγότερο κόστος από τις Πεπερασμένες Διαφορές. Βασική προϋπόθεση είναι το σύστημα να είναι εργοδικό (ergodic [1]), που σημαίνει πως για αρκετά μεγάλο χρόνο ολοκλήρωσης, η αντικειμενική συνάρτηση να είναι ανεξάρτητη των αρχικών συνθηκών του προβλήματος, ιδιότητα που χαρακτηρίζει τα περισσότερα φυσικά συστήματα, όπως τυρβώδεις ροές ρευστών. Αυτή η υπόθεση επιτρέπει τη 'σύγκριση' δύο τροχιών, μίας με τιμή μεταβλητής σχεδιασμού $\rho$ (τροχιά αναφοράς $\vec{u}_r$), και μίας άλλης με $\rho + \Delta\rho$ (σκιώδης τροχιά $\vec{u}$ της πρώτης), που έχουν διαφορετικές αρχικές συνθήκες. Την ύπαρξη σκιώδους τροχιάς για κάθε τιμή του $\rho$ εγγυάται το Λήμμα περί Σκίασης (Shadowing Lemma) [1]. Δύνανται να επιλεγούν αρχικές συνθήκες και ένας χρονικός μετασχηματισμός $\tau(t)$,




Σχήμα 2.2: *Lorenz '63* Η SD υπολογισμένη με τη Συνεχή Συζηγή Μέθοδο, για $\rho$ από 0 ως 100, για 20 τυχαίες αρχικές συνθήκες.




Σχήμα 2.3: *Lorenz '63* Η SD υπολογισμένη με Πεπερασμένες Διαφορές, για $\rho$ από 0 ως 100, για 20 τυχαίες αρχικές συνθήκες. Τα μπλε σημεία αντιστοιχούν σε χρόνο ολοκλήρωσης $T = 20$ ενώ τα κόκκινα σε $T = 2000$ μονάδες χρόνου.

έτσι ώστε οι δύο τροχιές να παραμένουν η μία 'κοντά' στην άλλη, για κάθε $0 \leq t \leq T$, επιλύοντας το παρακάτω πρόβλημα βελτιστοποίησης υπό περιορισμούς:

$$\min_{\vec{u},\eta} \frac{1}{2} \int_0^T \|\vec{u}\,(\tau(t)) - \vec{u}_r\|^2 + \alpha^2 \left(1 - \frac{d\tau}{dt}\right)^2 dt, \qquad s.t. \quad \frac{d\vec{u}}{d\tau} = \vec{f}(\vec{u}, \rho + \delta\rho), \qquad 0 < t < T \tag{2.3}$$

όπου το $\vec{f}(\vec{u}, \rho)$ συμβολίζει το δεξί μέρος της Εξ. 2.1. Η σταθερά $\alpha$ επιλέγεται ώστε τα δύο μέλη του ολοκληρώματος να είναι κοντινής τάξης μεγέθους. Διαφορίζοντας ως προς $\rho$, η Εξ. 2.3 μετασχηματίζεται σε:

$$\min_{\vec{v},\eta} \frac{1}{2} \int_0^T \|\vec{v}\|^2 + \alpha^2 \eta^2 \, dt, \qquad s.t. \quad \frac{d\vec{v}}{dt} = \frac{\partial \vec{f}}{\partial \vec{u}} \vec{v} + \frac{\partial \vec{f}}{\partial \rho} + \eta \vec{f}, \qquad 0 < t < T \tag{2.4}$$

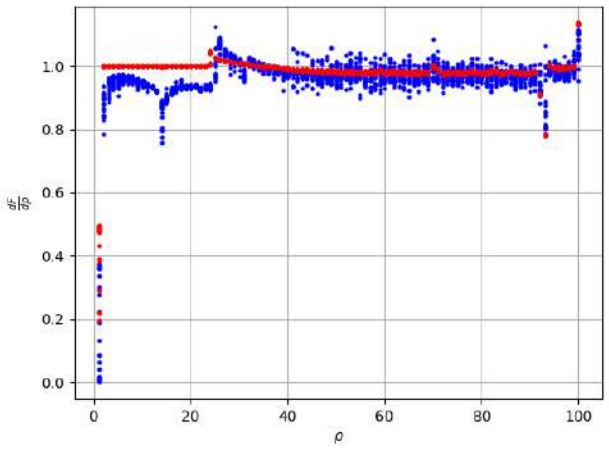
όπου $\vec{v} \equiv \frac{\partial \vec{u}}{\partial \rho}$. Οι εξισώσεις που προκύπτουν από τις συνθήκες Karush-Kuhn-Tucker (KKT) είναι:

$$\frac{d\vec{v}}{dt} = \frac{\partial \vec{f}}{\partial \vec{u}} \vec{v} + \frac{\partial \vec{f}}{\partial \rho} + \eta \vec{f} \tag{2.5α′}$$

$$\frac{d\vec{w}}{dt} = -\left(\frac{\partial \vec{f}}{\partial \vec{u}}\right)^T \vec{w} + \vec{v} \tag{2.5β′}$$

$$\eta = \frac{1}{\alpha^2} \vec{w}^T \vec{f} \tag{2.5γ′}$$

$$\vec{w}(0) = \vec{w}(T) = 0 \tag{2.5δ′}$$




Σχήμα 2.4: *Lorenz '63* Οι τιμές των SDs με 20 τυχαίες αρχικές συνθήκες για κάθε τιμή του $\rho$, υπολογισμένες με τη μέθοδο LSS. Τα μπλε σημεία αντιστοιχούν σε χρόνο ολοκλήρωσης $T = 20$, ενώ τα κόκκινα σε $T = 2000$.

και συγχωνεύονται σε μία μόνο συνήθη διαφορική δεύτερης τάξης, η οποία ενσωματώνει τις BCs με φυσικό τρόπο:

$$\frac{d^2\vec{w}}{dt^2} + \left[\left(\frac{\partial\vec{f}}{\partial\vec{u}}\right)^T - \frac{\partial\vec{f}}{\partial\vec{u}}\right]\frac{d\vec{w}}{dt} + \left[\frac{d}{dt}\left(\frac{\partial\vec{f}}{\partial\vec{u}}\right)^T - \frac{\partial\vec{f}}{\partial\vec{u}}\left(\frac{\partial\vec{f}}{\partial\vec{u}}\right)^T - \frac{1}{a^2}\vec{f}\vec{f}^T\right]\vec{w} - \frac{\partial\vec{f}}{\partial\rho} = 0,$$
$$\vec{w}(0) = \vec{w}(T) = 0 \tag{2.6}$$

Κατόπιν επίλυσης της Εξ. 2.6, μπορούν να υπολογιστούν για κάθε $t$ τα $\vec{v}$ και $\eta$. Έπειτα, η Εξ. 2.7 δίνει την παράγωγο ευαισθησίας:

$$\frac{\delta F}{\delta\rho} = \frac{1}{T}\int_0^T (v_z + \eta z)\ dt - \frac{1}{T^2}\int_0^T \eta\, dt \int_0^T z\, dt \tag{2.7}$$

Στο πλαίσιο αυτής της διπλωματικής αναπτύχθηκε μία Διακριτά Συμβατή διατύπωση της μεθόδου LSS, Discretely Consistent LSS (DCLSS), η οποία λαμβάνει υπόψη τη συμβατότητα των σχημάτων διακριτοποίησης των χρονικών παραγώγων που προκύπτουν στη συγχώνευση των Εξ. 2.5 για την παραγωγή της Εξ. 2.6, και παράγει αποτελέσματα βελτιωμένης ακρίβειας, ή και ορθά εκεί που αποτυγχάνει η LSS.

Το Σχ. 2.4 δείχνει τις SD για $\rho$ από 0 ως 100 και $\alpha = 30$, με 20 τυχαίες αρχικές συνθήκες για κάθε τιμή του $\rho$, που προκύπτουν από τη μέθοδο LSS. Είναι προφανές πως η LSS επιτυγχάνει να υπολογίσει τις σωστές SDs με μεγαλύτερη ακρίβεια από τις αντίστοιχες των Πεπερασμένων Διαφορών.

# Κεφάλαιο 3

# Αφομοίωση Δεδομένων

Η Αφομοίωση Δεδομένων (Data Assimilation, DA) [5, 3, 4, 6, 2] είναι μία οικογένεια μεθόδων που συνδυάζουν την πληροφορία από πειραματικά δεδομένα και μοντέλα που περιέχουν σφάλμα, ώστε να παράξουν αποτελέσματα με μαθηματικά αποδεδειγμένη υψηλότερη ακρίβεια, από εκείνη κάθε πηγής ξεχωριστά. Χρησιμοποιείται ευρέως σε τομείς όπως η μετεωρολογία και η ωκεανογραφία. Εδώ χρησιμοποιείται η μέθοδος του Εκτεταμένου Φίλτρου Kalman (Extended Kalman Filter, EKF) η οποία ενσωματώνει πειραματικά δεδομένα με σφάλμα κατά τη διάρκεια χρονικά μη-μόνιμης προσομοίωσης, βασισμένης σε μοντέλο το οποίο αναπόφευκτα εμπεριέχει σφάλμα, υπολογίζοντας μία κατάσταση συστήματος βελτιωμένης ακρίβειας.

Το μοντέλο $M$ λαμβάνει ως είσοδο την κατάσταση $\vec{u}$ του συστήματος στο χρονικό βήμα $k-1$, είτε είναι η αφομοιωμένη (δείκτης $a$) είτε προέρχεται από το μοντέλο (δείκτης $m$) στο προηγούμενο βήμα, και παράγει την κατάσταση στο τρέχον βήμα $k$. Εκείνη είτε θα χρησιμοποιηθεί αυτούσια στο επόμενο βήμα, είτε θα συνδυαστεί με πειραματικά δεδομένα (παρατηρήσεις, observations) $\vec{v}$ για να παραχθεί η αφομοιωμένη κατάσταση. Το μοντέλο περιγράφεται από την Εξ. 3.1:

$$\vec{u}_k^m = M\left(\vec{u}_{k-1}^a\right) \tag{3.1}$$

Οι παρατηρήσεις $\vec{v}$ δίνονται από την Εξ. 3.2, ως συνάρτηση της πραγματικής κατάστασης του συστήματος $\vec{u}^t$ (η οποία είναι πάντοτε άγνωστη), με τη βοήθεια ενός τελεστή $H$, που είναι υπεύθυνος για την πραγματοποίηση παρεμβολής ή μετασχηματισμού.

$$\vec{v} = H\left(\vec{u}^t\right) + \vec{e}^o \tag{3.2}$$

όπου $\vec{e}^o$ το σφάλμα που συνοδεύει την παρατήρηση των πειραματικών δεδομένων. Το μοντέλο επίσης υπόκειται σε σφάλμα, το οποίο περιγράφεται από τη σχέση:

$$\vec{u}_k^t = M\left(\vec{u}_{k-1}^t\right) + \vec{e}_{k-1}^m \tag{3.3}$$

Για ευκολία, χωρίς απώλεια της γενικότητας, τα σφάλματα που παρουσιάζονται σε

αυτήν την ανάλυση θεωρούνται πως ακολουθούν κανονική κατανομή με μέση τιμή μηδέν. Η αφομοιωμένη κατάσταση (assimilated state, δείκτης $a$), δίνεται από την Εξ. 3.4, ως συνάρτηση της παρατήρησης και της εξόδου του μοντέλου:

$$\vec{u}^{\,a} = \vec{u}^{\,m} + \mathbf{K}\left(\vec{v} - H\left(\vec{u}^{\,m}\right)\right) \tag{3.4}$$

όπου $\mathbf{K}$ είναι το EKF. Ο υπολογισμός του EKF αποτελεί τον στόχο αυτής της ανάλυσης, και γίνεται με βάση δύο παραδοχές [2]. Πρώτον, πως η αφομοιωμένη κατάσταση έχει μέση τιμή την άγνωστη πραγματική κατάσταση, επομένως έχει τυχαίο σφάλμα με μέση τιμή μηδέν, και δεύτερον πως η διασπορά (variance) του σφάλματος είναι η ελάχιστη από κάθε άλλο συνδυασμό της κατάστασης του μοντέλου και των παρατηρήσεων, καθιστώντας εκείνη τη βέλτιστη. Το EKF δίνεται από την Εξ. 3.5:

$$\mathbf{K} = \mathbf{C^m}\,\nabla\mathbf{H}\big|^T_{\vec{u}\,m}\left[\nabla\mathbf{H}\big|_{\vec{u}\,m}\,\mathbf{C^m}\,\nabla\mathbf{H}\big|^T_{\vec{u}\,m} + \mathbf{C^o}\right]^{-1} \tag{3.5}$$

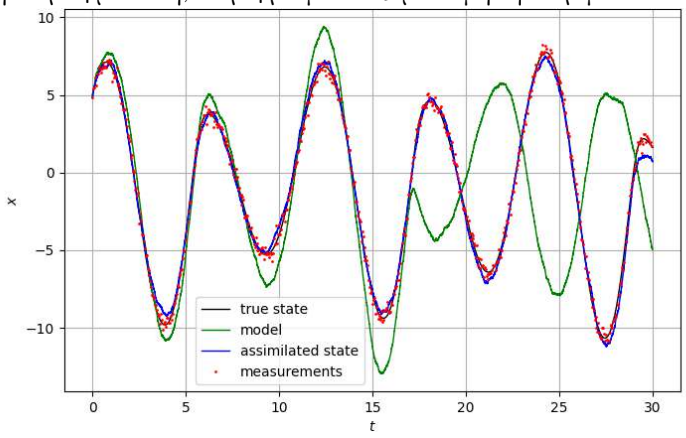
όπου $\mathbf{C^m}$ και $\mathbf{C^o}$ η (γνωστή) διασπορά του σφάλματος του μοντέλου και των παρατηρήσεων, αντιστοίχως. Στο Σχ. 3.1 παρατηρείται εφαρμογή της μεθόδου για το χαοτικό σύστημα του Rössler:

$$\frac{dx}{dt} = -y - z, \qquad x(0) = x_0 \tag{3.6α′}$$

$$\frac{dy}{dt} = x + ay, \qquad y(0) = y_0 \tag{3.6β′}$$

$$\frac{dz}{dt} = b + z(x - c), \qquad z(0) = z_0 \tag{3.6γ′}$$

όπου φαίνεται με πράσινο η κατάσταση που επιστρέφει το μοντέλο για $0 \leq t \leq 30$ χωρίς κάποια διόρθωση, με μαύρο η πραγματική κατάσταση του συστήματος, με κόκκινες κουκίδες οι πειραματικές μετρήσεις στις χρονικές στιγμές που είναι διαθέσιμες, ενώ με μπλε φαίνεται η αφομοιωμένη κατάσταση. Μπορεί κανείς να παρατηρήσει πως το μοντέλο μόνο του αρχίζει πολύ γρήγορα να συσσωρεύει μη-αποδεκτό σφάλμα, και αποτυγχάνει να περιγράψει το σύστημα. Αντιθέτως, η αφομοιωμένη κατάσταση σχεδόν ταυτίζεται με την πραγματική, περιγράφοντας με υψηλή ακρίβεια το σύστημα



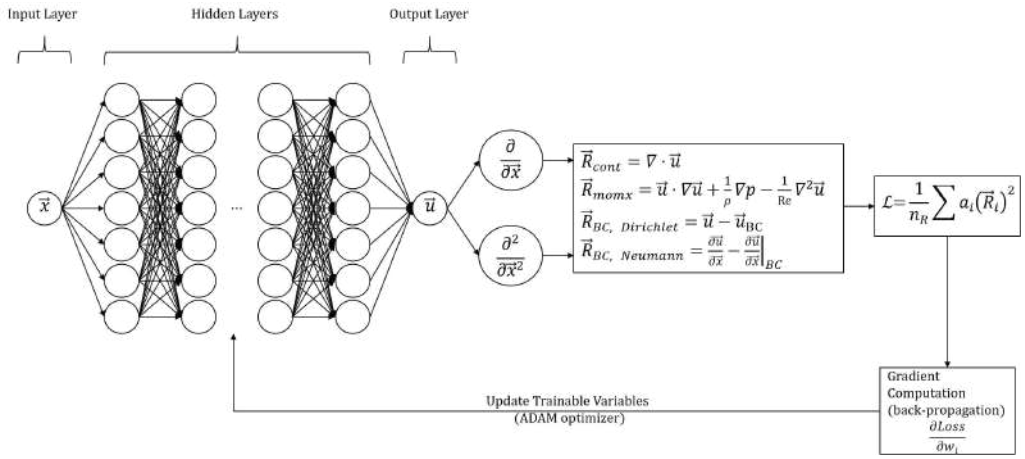

Σχήμα 3.1: *Rössler*: Η πραγματική (μαύρο), η αφομοιωμένη (μπλε), η κατάσταση του μοντέλου (πράσινο) και οι παρατηρήσεις για την $x$ συνιστώσα του συστήματος.

# Κεφάλαιο 4

# Ενημερωμένα από τη Φυσική των Ροών Νευρωνικά Δίκτυα

Τα Ενημερωμένα από τη Φυσική των Ροών Νευρωνικά Δίκτυα (Physics-Informed Neural Networks, PINNs) είναι νευρωνικά δίκτυα που λύνουν διαφορικές εξισώσεις ελαχιστοποιώντας μια συνάρτηση απωλειών που περιλαμβάνει τα υπόλοιπα των εξισώσεων και τις συνοριακές ή/και αρχικές συνθήκες. Σε αυτή τη διπλωματική εργασία βασίζονται στην αρχιτεκτονική των Βαθέων Νευρωνικών Δικτύων (Deep Neural Networks, DNNs). Σε αντίθεση με τις συμβατικές αριθμητικές μεθόδους, οι λύσεις που προσφέρουν τα PINNs είναι αναλυτικές, καθιστώντας μη-αναγκαία κάθε παρεμβολή. Οι παράγωγοι που χρησιμοποιούνται στα υπόλοιπα, υπολογίζονται μέσω αυτόματης διαφόρισης αποφεύγοντας τη χρήση σχημάτων διακριτοποίησης. Παρ΄όλα αυτά, τα PINNs απαιτούν πολλές πράξεις, με αποτέλεσμα να αυξάνεται ο χρόνος επίλυσης σε σχέση με τις συμβατικές μεθόδους.

Εξετάζονται δύο εφαρμογές: πρώτον, η βελτιστοποίηση ψευδο-1D ροής μέσω ξεχωριστών PINNs που επιλύουν το πρωτεύον και το συζυγές πρόβλημα, που χρησιμοποιούνται σε πρόβλημα αντίστροφου σχεδιασμού. Δεύτερον, η λύση των εξισώσεων Navier-Stokes για 2D στρωτή, ασυμπίεστη ροή σε αγωγό μεταβαλλόμενης διατομής, όπου τα αποτελέσματα συγκρίνονται με τον GPU-επιταχυνόμενο CFD κώδικα PUMA του PCOpt/NTUA. Ο αλγόριθμος εκπαίδευσης που χρησιμοποιείται φαίνεται στο Σχ. 4.1, εφαρμοσμένος στο 2D πρόβλημα της δεύτερης εφαρμογής.




Σχήμα 4.1: Διάγραμμα της διαδικασίας εκπαίδευσης του PINN που λύνει για στρωτή ασυμπίεση 2D ροή.

## 4.1 Ψευδο-1D Ροή

Στην πρώτη εφαρμογή πραγματοποιείται επίλυση του ευθέος και συζυγούς προβλήματος ψευδο-1D, ασυμπίεστης μόνιμης ροής μέσω αγωγού μεταβλητής διατομής, και επιλύεται πρόβλημα αντίστροφου σχεδιασμού που αποσκοπεί στην απόκτηση καθορισμένης κατανομής πίεσης

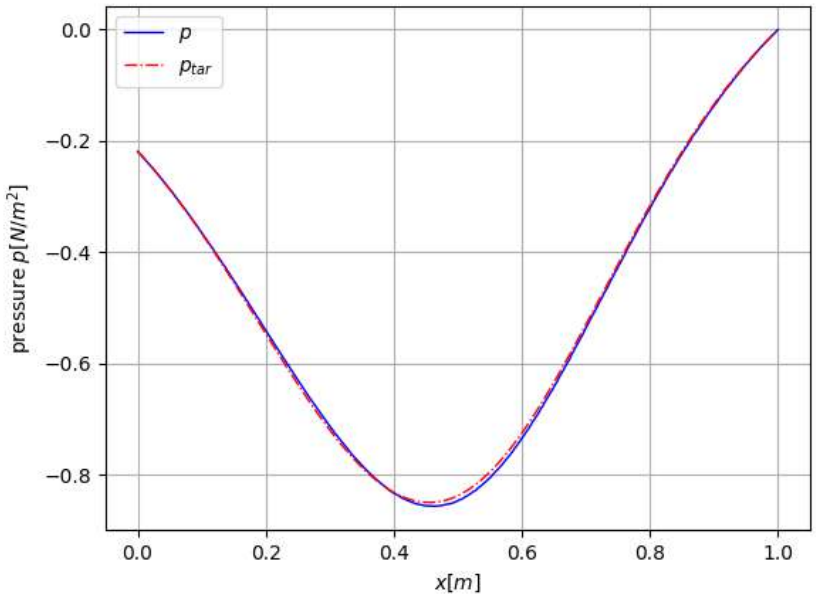
$$\begin{aligned} \frac{d(vS)}{dx} = 0, \quad u(x=0) &= u_{BC} = 1\, m/s \\ \rho v \frac{dv}{dx} + \frac{dp}{dx} = 0, \quad p(x=1) &= p_{BC} = 0\, N/m^2 \end{aligned} \tag{4.1}$$

Οι συζυγείς εξισώσεις βρέθηκαν με τη Συνεχή Συζυγή Μέθοδο (Continuous Adjoint Method), έτσι ώστε να είναι εφαρμόσιμες στο συνεχές PINN. Στο Σχ. 4.2 φαίνεται η σύγκριση της κατανομής πίεσης που βρέθηκε με τη βελτιστοποίηση, με την κατανομή-στόχο (μπλε και κόκκινη γραμμή αντίστοιχα). Είναι εμφανές πως η βελτιστοποίηση ήταν επιτυχής, καθώς οι γραμμές σχεδόν ταυτίζονται.

## 4.2 2D Ροή

Στη δεύτερη εφαρμογή επιλύεται μόνιμη, 2D, στρωτή ροή ασυμπίεστου ρευστού μέσω αγωγού μεταβλητής διατομής, που διέπεται από τις εξισώσεις Navier-Stokes, με $Re = 120$:

$$\begin{aligned} \nabla \cdot \vec{u} &= 0 \\ \vec{u} \nabla \vec{u} - \frac{1}{Re} \nabla^2 \vec{u} + \frac{1}{\rho} \nabla p &= 0 \end{aligned} \tag{4.2}$$

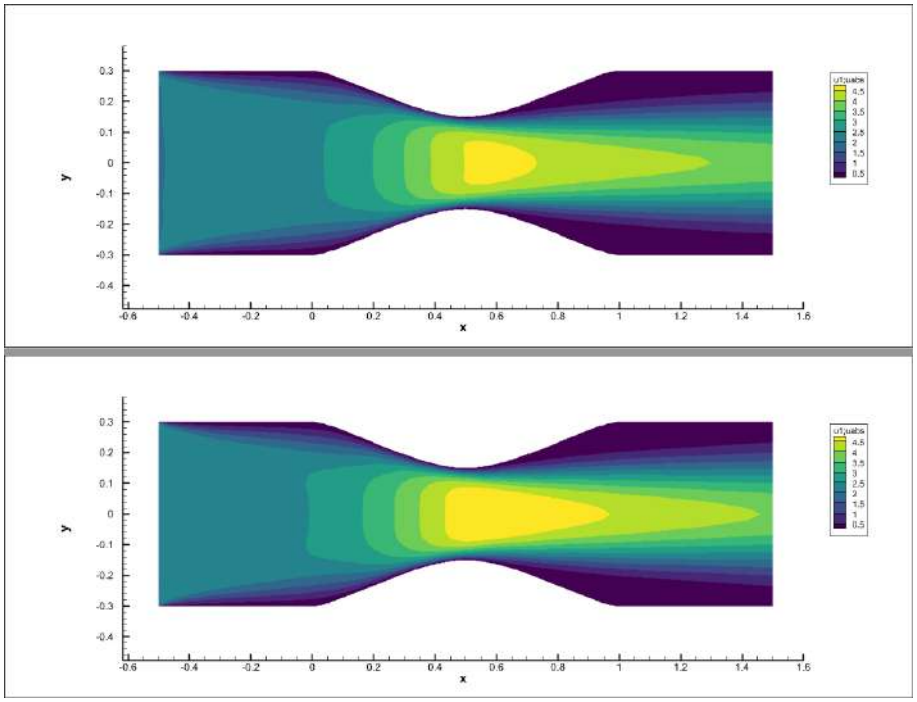



Σχήμα 4.2: Η κατανομή πίεσης από το πρόβλημα βελτιστοποίησης (μπλε) σε σύγκριση με την κατανομή-στόχο (κόκκινο).

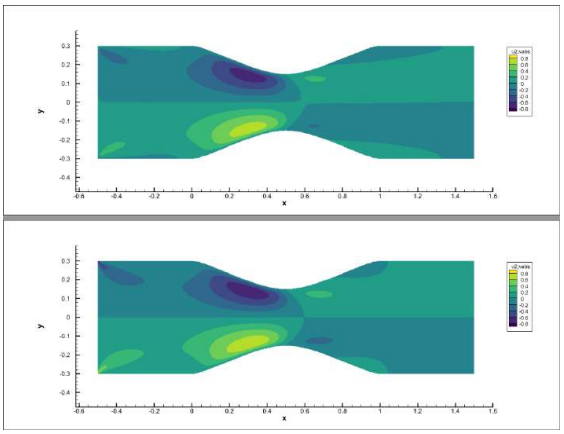
Οι οριακές συνθήκες που χρησιμοποιήθηκαν είναι:

1. **Είσοδος:** $u = 1$ και $v = 0$
2. **Έξοδος:** $p = 0$ (πίεση αναφοράς) και $\frac{\partial u}{\partial x} = \frac{\partial v}{\partial x} = 0$
3. **Στερεά σύνορα:** $u = v = 0$ (συνθήκη μη-ολίσθησης)

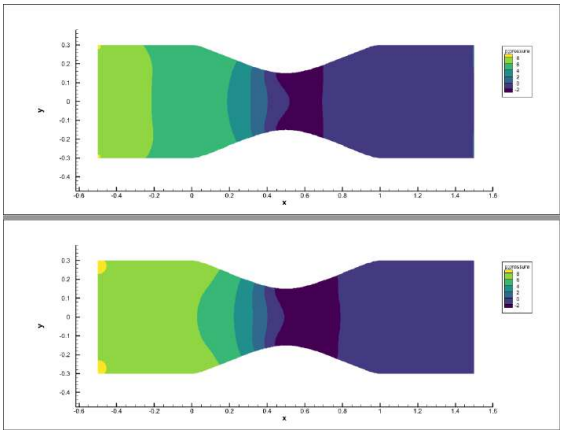
Στα Σχ. 4.3, 4.4 και 4.5 φαίνονται τα πεδία οριζόντιας ταχύτητας, κατακόρυφης ταχύτητας και πίεσης που προέκυψαν από τη λύση του PINN (πάνω εικόνες), σε σύγκριση με τη λύση του PUMA. Είναι εμφανές πως το PINN κατάφερε να λύσει με ικανοποιητική ακρίβεια τη ροή.



Σχήμα 4.3: Οριζόντια ταχύτητα της λύσης του PINN (πάνω) και του PUMA (κάτω).



Σχήμα 4.4: Κατακόρυφη ταχύτητα της λύσης του PINN (πάνω) και του PUMA (κάτω).



Σχήμα 4.5: Πίεση της λύσης του PINN (πάνω) και του PUMA (κάτω).

# Bibliography


[1] Blonigan, P.J.: New Methods for Sensitivity Analysis of Chaotic Dynamical Systems. Master's thesis, Massachusetts Institute of Technology (2013)

[2] G.A.Terejanu: Extended kalman filter tutorial. `https://homes.cs.washington.edu/~todorov/courses/cseP590/readings/tutorialEKF.pdf` (2014), center for Turbulence Research, Proceedings of the Summer Program 2018, arXiv:1902.11112 [cs.CE]

[3] G.Evensen: Data Assimilation. Springer, Bergen, Norway

[4] Law, K., Stuart, A., Zygalakis, K.: Data Assimilation. Springer

[5] N.J.Kutz: Data-Driven Modeling  Scientific Computation. Oxford University Press, Great Clarendon Street, Oxford, OX2 6DP, United Kingdom

[6] S.K.Park, Zupanski, M.: Principles of Data Assimilation. Cambridge University Press

[7] Wang, Q.: Forward and adjoint sensitivity computation of chaotic dynamical systems. `https://arxiv.org/abs/1202.5229` (2012)