Imperial College London

Aeronautical Engineering

Final Year Project

Extracting Equations From Noisy Data Using Machine Learning

Academic Supervisor:
Dr G. Papadakis

Trieu ho
CID: 01057749

May 22, 2019

Abstract

In this report we propose the use of the Sequential Thresholded Least Squares (STLS) [1] methodology to identify the dynamics of a low-dimensional model on turbulent shear flow, more specifically sinusoidal shear flow [2, 3]. Whereby the states of the model are estimated from collected velocity, and the minimum number of sensor points for data collection was determined to be 5, with 3-dimensional velocities collected at each point. This enabled the capture the underlying partial differential equations (PDEs) of the sinusoidal shear flow, with minimal sensor points. There is a fractal dependence for lifetimes of turbulence on the perturbation initial energy, E, and Reynolds number of the flow Re. The error of the coefficients produced by STLS compared to the exact coefficients was found to exponentially decay with the lifetime of turbulence. This in turn suggests a fractal behaviour of the accuracy of the STLS methodology on initial energy and Re. The low-dimensional model on turbulent shear flow has a positive Lyapunov exponent and as a result, small differences in initial conditions or identified coefficients will result in errors growing exponential with time until saturation.

Contents

List of Figures

List of Tables

1 Introduction

In science and engineering, many of the models used to describe underlying dynamics of a system are developed from a theoretical basis, using first principles, conservation laws or physical principles. However, there are many complex systems that are yet to be described by an underlying model or even characterised by a suitable choice of variables, for instance in ecology, finance and neuroscience [1]. With the improvements in technology, more specifically better data handling capabilities, lower cost of sensors, and improving computer processors, large data is becoming more abundant [4]. This has prompted the increasing popularity in the data-driven discovery of underlying partial differential equations of dynamical systems. The use of genetic programming in determining underlying nonlinear dynamical systems from large data was proposed by [5], and since has inspired other methodologies such as the SINDys [1] which uses the Sequential Thresholded Least Squares (STLS) algorithm. The STLS algorithm is the core methodology used in this paper in conjunction with a prerequisite conversion of velocity data into states data.

In fluid dynamics, the Navier-Stokes Equation characterises viscous flows and captures important flow behaviours that have revolutionised aviation and space exploration [6]. The direct numerical simulation (DNS) of variations of the Navier-Stokes have lead to many discoveries in turbulent flows. However, even with the development of faster supercomputers, DNS of the Navier-Stokes Equation is very costly. With the number of floating-point operations required to run the simulation proportional to number of mesh points and time steps, and further grows as the cube of the Reynolds number [7]. As such, many simplified models have been made in an attempt to describe certain flow behaviours. These simplified models usually find inspiration from observations and data collected in experiments or DNS. Numerical simulations and experimental data show that the dynamics of turbulent shear flow is spatially and temporally complicated up to the onset of turbulence [8]. Due to its geometric simplicity, many low-dimensional models are developed for the sinusoidal shear flows in order to better understand transition from laminar to turbulence. The sinusoidal shear flow consists of two walls with no-slip boundaries and the fluid in between experiences a sinusoidal body force. This results in a sinusoidal laminar profile. It was suggested that downstream vortices play a major role in transition, whereby the unstable modes of the streaks are recycled as downstream vortices. However, the constraints posed by the form of the Navier-Stokes limits the interaction between modes and makes the development of low-dimensional models more difficult [2]. A 9-states low dimensional model was developed by Moehlis et al [2, 3], to describe the flow behaviours of a sinusoidal shear flow. This 9-states model builds on the work by Waleffe [9, 10], where the Fourier modes are extracted from proper orthogonal decomposition of numerical data [11]. The main components of the modes include a basic mean velocity profile, downstream vortices, streaks, instability of the streaks, modes arising from advection of previous modes and a modification to the mean velocity profile to account for changes caused by fluctuations [2].

The STLS algorithm is used in this paper in conjunction with a preliminary induction of states data from collected velocity data, as seen in Section 3.1. The ultimate aim of this paper is to showcase a methodology in which wind tunnel data can be collected and the coefficients of the underlying partial differential equation can be found for a set of defined modes. This would enable the testing of different Fourier modes and would also capture the complex coupling between modes, all whilst respecting the constraints posed by the Navier-Stokes.

The outline of this paper is as follows. The paper begins with an outline of previous works and a generic problem set-up is highlighted in Section 2. This is followed by the implementation of the STLS methodology on the Lorenz system in Section 2.3, which is a simple 3-dimensional example to which the STLS methodology can be easily explained. The STLS method is described in Algorithm 1. In section 3 the application of the STLS methodology is implemented on the low dimensional model for turbulent shear flow. An extra step is added to the STLS, whereby the states are estimated from velocity data prior to the identification of the nonlinear ODEs, this forms the preSTLS method seen

in Algorithm 2. In section 4, the performance of the preSTLS method is investigated for different flow characteristics. Finally, this paper ends with concluding remarks and highlights future research ideas.

The project aims and objectives are summarised below

1. Implementation of the STLS Methodology [1]
   (a) Research and compare to other regression methodologies
2. Explore different noise mitigation methodologies
   (a) Locally weighted scatter plot smooth
   (b) Bayesian wavelet denoising
3. Application of STLS on Lorenz system
4. Application preSTLS on low-dimensional model for turbulent shear flow [2, 3]
   (a) Extract states data from minimal velocity extraction points
   (b) Explore effects of Reynolds number on the performance of the preSTLS algorithm
   (c) Explore effects of domain size on the performance of the preSTLS algorithm
   (d) Explore effects of turbulent lifetime on accuracy of the preSTLS in identifying the underlying PDE

2 Background and literature review

Throughout this paper, the terms *noise* and *sparsity* will be used. Therefore, it is convenient to defined these terms at an early stage.

Noise can be defined as any disturbances or fluctuations to the measured original information [12]. The removal of noise, also referred to as denoising, is a vastly researched topic and plays a very important role in image and data processing [13]. Throughout this paper, noise is defined as additive white Gaussian with a given scalar signal-to-noise ratio (snr) defined in decibels. White Gaussian noise is typical for most instrument-measured data and arises from natural sources such as thermal vibration [14]. There are of course a multitude of different noise distributions that can arise in real-life and different ways to deal with them, these are well explained in [12], however, in this paper we will only look at the effects of white Gaussian noise on the algorithm. The white Gaussian noise is generated using the MATLAB **awgn**(*input*, *snr*, *signalpower*) function, with the input signal *input* specified as a vector for each state. The *snr* is the ratio of signal power P_{signal} to the noise power P_{noise}, or the ratio of their squared amplitudes A. A signal to noise ratio of greater than 1:1 (0 dB) implies there is more signal than noise. The *signalpower* input is defined as 'measured', in which the function measures the power of *input* in dBW before noise is added. For context, a signal-to-noise ratio of 20 dB means that the signal has 100 times the power of the noise, or 10 times the amplitude.

$$snr = \frac{P_{signal}}{P_{noise}} = \left(\frac{A_{signal}}{A_{noise}}\right)^2 \tag{1}$$

or in decibels

$$snr_{dB} = 10log_{10}\left(\frac{P_{signal}}{P_{noise}}\right) = 20log_{10}\left(\frac{A_{signal}}{A_{noise}}\right) \tag{2}$$

Sparsity in the context of machine learning refers to the trade-off between goodness-of-fit and the complexity of the final results [15]. Typically, regression methodologies return coefficient matrices which corresponds to the importance of each term in a library of trial functions, and the degree of sparsity is simply the number of zero terms in the aforementioned matrices. Generally, many regression methodologies such as least squares or ridge regression, return a non-sparse coefficient matrix, whereby the non-important terms are small but not zero. This results in over-fitting the data and can make the identified PDEs complex. As such, regression methods that induces sparsity is desired when determining the underlying dynamical equations, as a sparse representation will give the simplest form of the equation without much compromise to the accurate representation of the data. The degree of sparsity in this paper is chosen at the Pareto front, whereby plotting the number of non-zero terms against the threshold parameter reveals an L-curve and the chosen threshold is selected at the 'kink' or 'elbow' of the L-curve. This ensures that the algorithm returns a sparse representation of the underlying dynamical system without over-fitting the data.

2.1 Methodology

The ultimate aim of paper is to define a methodology in which one can collect real-life data of non-linear dynamical systems, such as such wind tunnel velocity data for sinusoidal shear flow, followed by the mitigation of the noise in the data, and putting it through the regression algorithm to finally outputting the governing dynamical equations. The technical setup is the same as that given in [1], but for completeness it will be outlined in this paper.

Consider the non-linear dynamical system of the form:

$$\frac{d}{dt}\mathbf{X}(t) = \mathbf{f}(\mathbf{X}(t)) \tag{3}$$

Where $\mathbf{X}(t) = [x_1(t) \quad x_2(t) \quad ... \quad x_n(t)]^T \in \mathbb{R}^n$ is the state of the system at time t. The non-linear function $\mathbf{f}(\mathbf{X}(t))$ is the dynamical constraints that defines the equation of motion of the system [1]. In other words, Equation 3 defines the relationship between the time derivative of the states and the non-linear function of states. In general we can write $\mathbf{f}(\mathbf{X}(t))$ in terms of a library of candidate functions $\mathbf{\Theta}[\mathbf{X}(t)]$, multiplied by the corresponding coefficient matrix $\mathbf{\Xi} = [\xi_1 \quad \xi_2 \quad ... \quad \xi_n]$. Where ξ_i corresponds to the coefficient vector of the i^{th} state.

$$\frac{d}{dt}\mathbf{X}(t) = \mathbf{\Theta}[\mathbf{X}(t)]\mathbf{\Xi} \tag{4}$$

In this paper, we sought to define a set of candidate trial functions $\mathbf{\Theta}[\mathbf{X}(t)]$ and solve regression problems to find the coefficient matrix $\mathbf{\Xi}$. We will assume that our choice of $\mathbf{\Theta}[\mathbf{X}(t)]$ is dense enough to return a sparse representation of the underlying PDE.

2.1.1 Data Collection

We begin by sampling data from the dynamical system at different times $t_1, t_2, ..., t_m$. The only required data to perform regression on Equation 4 are the states $\mathbf{X}$ and its time derivative $\dot{\mathbf{X}}$. The collected data are then arranged in the following matrices:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^T(t_1) \\ \mathbf{x}^T(t_2) \\ \vdots \\ \mathbf{x}^T(t_m) \end{bmatrix} = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \cdots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \cdots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \cdots & x_n(t_m) \end{bmatrix} \tag{5}$$

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{\mathbf{x}}^T(t_1) \\ \dot{\mathbf{x}}^T(t_2) \\ \vdots \\ \dot{\mathbf{x}}^T(t_m) \end{bmatrix} = \begin{bmatrix} \dot{x}_1(t_1) & \dot{x}_2(t_1) & \cdots & \dot{x}_n(t_1) \\ \dot{x}_1(t_2) & \dot{x}_2(t_2) & \cdots & \dot{x}_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{x}_1(t_m) & \dot{x}_2(t_m) & \cdots & \dot{x}_n(t_m) \end{bmatrix} \tag{6}$$

Ideally, both are sampled directly, but in reality data collection could be costly and only the state data is collected, in such case the time derivative can be approximated using finite difference methodologies. However, noise in the states can hugely affect its time derivative and cause the STLS algorithm to fail. Therefore, different noise mitigation methodologies are explored in Section 2.1.2. The STLS algorithm used in this paper assumes that the states are in a sensible coordinate system, where it would return a sparse coefficient matrix given the trial functions. If not sensibly chosen, the coordinate system will result in failure of the STLS algorithm. This was explored in the supplementary material of [1].

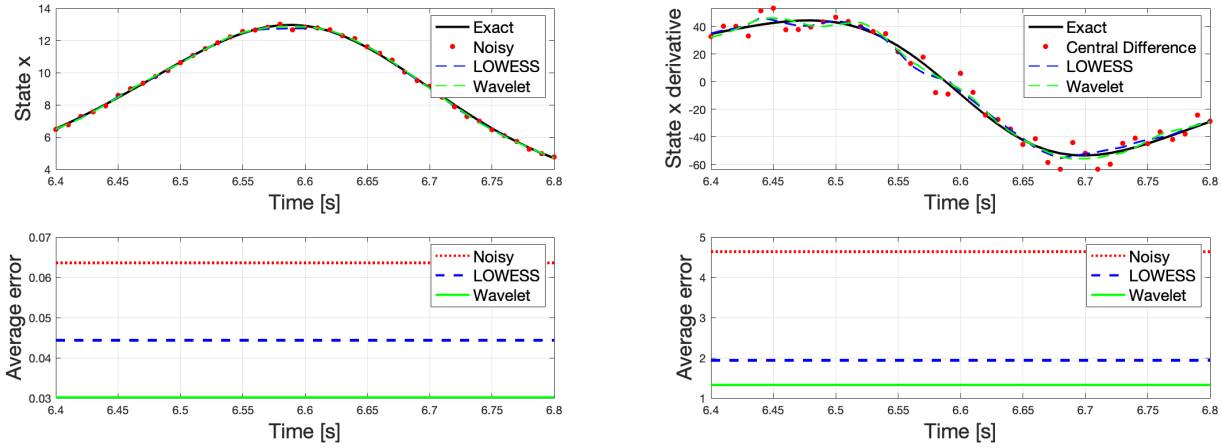
2.1.2 Preliminary Data Noise Reduction

Noise mitigation is a huge area of research and has many application in image reconstruction [16]. The noise that will most likely arise from the type of data that will be used in this paper e.g. wind tunnel data is white Gaussian noise, and the level of noise greatly impacts the accuracy of the algorithms.

For the Lorenz example, we will assume that only the noisy states data is available and the derivative is estimated using finite differences of the denoised states, see Section 2.3. For the low-dimensional shear

flow model example, we will assume that we only have noisy velocity data (note that the velocities are not the states). This noisy velocity data is denoised before estimating the states using least squares and the state derivative is estimated using central difference, see Section 3. This will provide good scenarios to test the robustness of the algorithms.

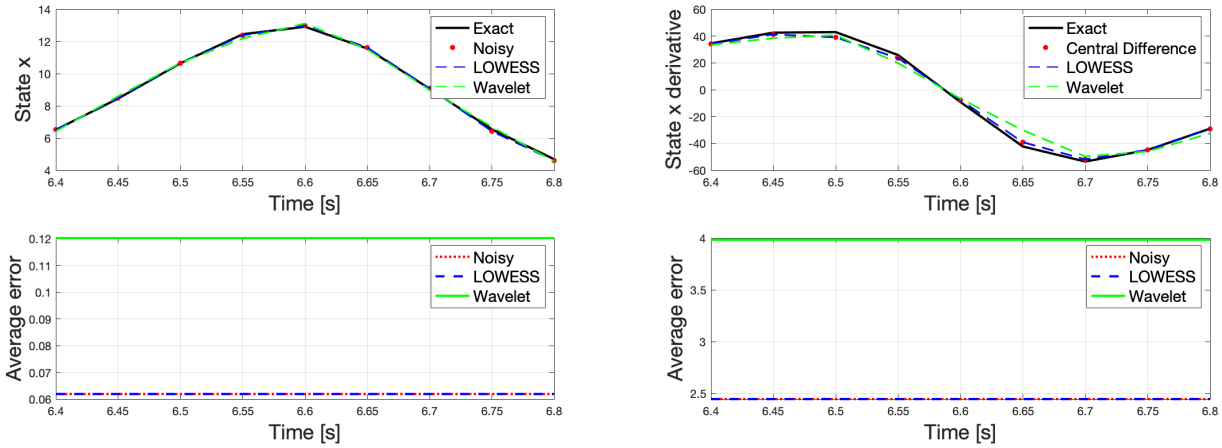
A comparison between different denoising methodologies was conducted, namely, Bayesian wavelet denoising (**wdenoise**)[17] from the MATLAB wavelet toolbox, and locally weighted scatter plot smooth (**lowess**) from the MATLAB curve-fitting toolbox [18]. The Bayesian denoising in the wavelet domain method uses a threshold rule from assuming that the input measurements have independent prior distributions given by a mixture model. The input measurements are used to estimate the weight in the mixture model, and the method usually works better with more samples. The default MATLAB settings for **wdenoise** was used. The **lowess** denoising method uses locally weighted linear regression to smooth data. Whereby the regression weight function for the data points are contained within a span. Furthermore, the *Robust* option was used to filter outliers, which is useful for the high noise cases. The input span for **lowess** was varied for different Δt.



(a) State values

(b) State derivative values

Figure 1: Comparison between different denoising methodologies for state x of the Lorenz system at a snr of 40 dB, with a fine sampling time step of $\Delta t = 0.01$.



(a) State values

(b) State derivative values

Figure 2: Comparison between different denoising methodologies for state x of the Lorenz system at a snr of 40 dB, with a coarse sampling time step of $\Delta t = 0.05$.

In Figures 1 and 2, the average error is the absolute difference between the between the noisy/denoised trajectory and the exact trajectory, averaged across the simulation time. The performances of the denoising methodologies are dependent on the sampling time step, whereby for a fine sampling time of $\Delta t = 0.01s$ is best handled with the Wavelet denoising methodology. However, for a coarse sampling time step of $\Delta t = 0.05s$, the application of the wavelet denoising methodology is adverse, and results in larger error than the noisy data, as such the **lowess** denoising is better in this case. For the Lorenz example, a sampling time step of $\Delta t = 0.01s$ is used and for the low-dimensional shear flow example a coarse sample time step of $\Delta t = 0.05$ is used. Therefore, wavelet denoising is used for the Lorenz example and **lowess** is used for the low-dimensional shear flow example.

2.1.3 Candidate Function Selection

Once the necessary noise filters are applied to the state matrix and its time derivative, we define a library of trial functions, $\mathbf{\Theta}$, which consists of functions that we believe will capture the underlying dynamics of a system. This library can consist of constants, polynomials, trigonometric functions and others. For instance,

$$\mathbf{\Theta}(\mathbf{X}) = [\quad \mathbf{1} \quad \mathbf{X} \quad \mathbf{X}^{p_2} \quad \mathbf{X}^{p_3} \quad ... \quad sin(\mathbf{X}) \quad cos(\mathbf{X}) \quad ... \quad] \tag{7}$$

The higher order polynomials such as $\mathbf{X}^{p_2}$ is the quadratic combinations of the state defined as

$$\mathbf{X}^{P_2} = \begin{bmatrix} x_1^2(t_1) & x_1(t_1)x_2(t_1) & \cdots & x_2^2(t_1) & x_2(t_1)x_3(t_1) & \cdots & x_n^2(t_1) \\ x_1^2(t_2) & x_1(t_2)x_2(t_2) & \cdots & x_2^2(t_2) & x_2(t_2)x_3(t_2) & \cdots & x_n^2(t_2) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_1^2(t_m) & x_1(t_m)x_2(t_m) & \cdots & x_2^2(t_m) & x_2(t_m)x_3(t_m) & \cdots & x_n^2(t_m) \end{bmatrix} \tag{8}$$

In general, if we assume that the library of trial function is sufficiently rich, then the PDE should be well-represented by Equation 4 with a sparse coefficient matrix Ξ [1]. Choosing a sufficiently rich library of candidate function might not have a clear methodology, but intuition can be used in some physical systems or assumptions can be drawn from experimental observations. Furthermore, trigonometric functions can be approximated using its Taylor expansion, so choosing polynomial trial function would be sufficient to capture trigonometric functions. The choices of trial functions are explored in more detail in the supplementary material for [1].

2.2 Regression Methodologies

In statistics, regression methodologies aim at estimating the relationships between variables, of that between a dependent variable and one or more independent variables [19]. Depending on the application, some regression methodologies are more favourable than others, with requirements such as sparsity for compressed sensing [20] or efficiency with large data sets. Some of the common regression methods are highlighted below, with the STLS method shown in Algorithm 1. The regression methods discussed below utilises both L^1 and L^2 norms of vectors, so for convenience these will be defined. For a vector $\mathbf{x} = [x_1 \quad x_2 \quad ... \quad x_n]^T$, the L^1 (left) and L^2 (right) norms are defined as

$$||\mathbf{x}||_1 = \sum_{i=1}^{n} |x_i| \qquad ||\mathbf{x}||_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2} \tag{9}$$

2.2.1 Least Squares Regression

One of the simplest regression method is the Least Squares Regression, which aims at minimising the L^2 norm ,and is the basis for the other regression methods discussed briefly in later sections of this paper. In the context of Equation 4, the least squares problem is defined by

$$\xi_i = argmin_{\xi_i} ||\mathbf{\Theta}\xi_i - \dot{\mathbf{x}}_i||_2 \qquad for \quad i = 1, 2, ..., n \tag{10}$$

This least squares problem has to be solved for each state independently, thus would be challenging for complex systems with a high number of states. Subsequently a matrix of coefficients Ξ can be found for a set of candidate function library Θ and in general would be non-sparse, as a result the least squares methodology tends to over-fit the data. Furthermore, the least squares methodology does not deal too well with collinearity. This is a phenomenon whereby one candidate function can be linearly predicted from others to a satisfactory degree of accuracy. In this case, the final extracted coefficients can change drastically when there are changes to the data. High collinearity usually arises when the candidate function library is large compared to the number of observations.

2.2.2 Ridge Regression

To deal with problems that arises from collinearity, regularizers are often used to put constraints to the least squares problem. One such regularizer is the Tikhonov regularization, which is an L^2 regularisation term. This regularized least square problem is known as Ridge regression and is defined by:

$$\xi_i = argmin_{\xi_i} ||\mathbf{\Theta}\xi_i - \dot{\mathbf{x}}_i||_2 + \lambda||\xi_i||_2 \qquad for \quad i = 1, 2, ..., n \tag{11}$$

With the least squares solution, the variance is minimised but this will lead to high bias. However, Ridge regression generally yields predictions than least squares, as it compromises between bias and variance. In general, Ridge regression does not induce sparsity in the solution, as it only shrinks the value of non-important terms but never equated to zero.

2.2.3 LASSO Regression

The least absolute shrinkage and selection operator (LASSO) is similar to the Ridge regression, which starts with the least square solution but this time with an L^1 regularizer.

$$\xi_i = argmin_{\xi_i} ||\mathbf{\Theta}\xi_i - \dot{\mathbf{x}}_i||_2 + \lambda||\xi_i||_1 \qquad for \quad i = 1, 2, ..., n \tag{12}$$

The LASSO method is good if sparsity is desired in the coefficient matrix as the unimportant terms are equated to zero. However, the LASSO method, much like the least squares, is highly influenced by collinearity. Furthermore, for large candidate function libraries, this method is highly inefficient in terms of computational costs.

2.2.4 Sequential Thresholded Least Squares

The methodology used in [1] is the sequential thresholded least squares (STLS). Whereby the method begins with the least squares solution as seen in Equation 10, followed by a hard threshold of the results, in which terms smaller than the threshold parameter, λ, are equated to zero. Subsequently the least squares solution is recalculated for the non-zero terms and this repeats until convergence. Algorithm 1 outlines the main steps in the STLS methodology.

Algorithm 1: Sequential thresholded least squares

function$[\xi] = \mathbf{STLS}(\boldsymbol{\Theta}, \dot{\mathbf{x}}, \lambda)$
$\xi_0 = \boldsymbol{\Theta} \setminus \dot{\mathbf{x}}$; % *initial guess of* ξ_0 *from least squares*
Error = 1;
while Error $> 10E-5$ **do**
  $\xi_{i+1} = \xi_i$;
  $smallinds = find(abs(\xi_i) < \lambda)$;
  $\xi_{i+1}(smallinds) = 0$;
  $biginds \sim= smallinds$;
  $\xi_{i+1}(biginds) = \boldsymbol{\Theta}(biginds) \setminus \dot{\mathbf{x}}$; % *least squares of rank reduced problem*
  Error = norm$(\xi_{i+1} - \xi_i)$;

This method is very powerful for a large candidate function libraries, as the rank of the matrix is reduced at each iteration and convergence is satisfied quickly. In addition, the sparsity of the final matrix of coefficient is only dependent on the threshold parameter, λ, so the degree of sparsity can be chosen simply by select an appropriate λ. For examples in this paper, λ is chosen at the Pareto front. Therefore, the primary regression method used in this paper is the STLS, whereby sparsity and fast convergence are achieved. However, an additional least squares step is added to this method as seen in Algorithm 2 for the low-dimensional shear flow example in Section 3.

2.3 Example 1: Lorenz System

The Lorenz system is a chaotic system that is ill-conditioned. The simplicity of the PDE and low number of states makes this example ideal for showcasing how the STLS algorithm works, before we move to the more complicated example provided by the low-dimensional turbulent shear flow in Section 3. To replicate the data collection process, the system given by Equations 13(a-c) is solved numerically using the MatLab ODE45 function. With these calculated states data, Gaussian noise is added at various snr values. In this example, since the run time is short, a fine sampling time step of $\Delta t = 0.01s$ is used. As discussed in Section 2.1.2, Bayesian wavelet denoising is most suitable in this case.

$$\dot{x} = \sigma(y - x) \tag{13a}$$
$$\dot{y} = x(\rho - z) - y \tag{13b}$$
$$\dot{z} = xy - \beta z \tag{13c}$$

Where the states are defined as $\mathbf{X} = [x \quad y \quad z]^T$, and we will use the example of $\sigma = 10, \quad \beta = 8/2, \quad \rho = 28$ and initial conditions $[x(0) \quad y(0) \quad z(0)]^T = [-8 \quad 7 \quad 27]^T$.

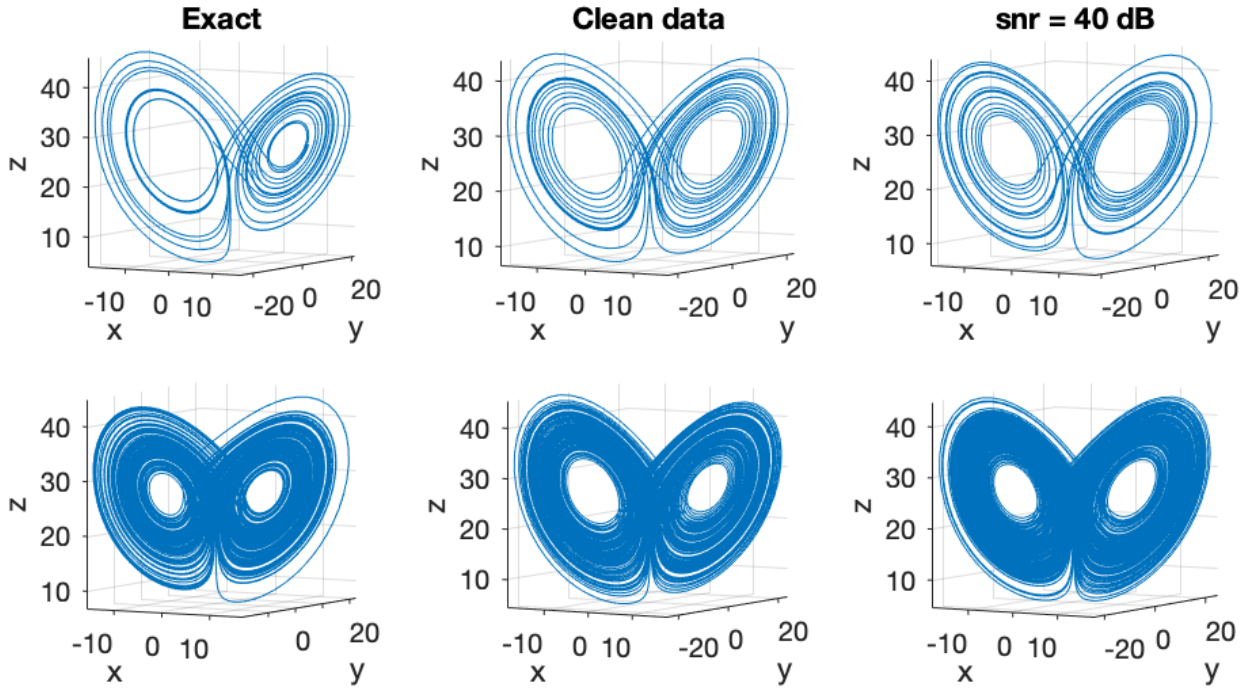


Figure 3: Lorenz trajectory of the exact system (left), the identified system with no noise (middle) and the identified system with a signal-to-noise ratio of 40 dB (right). Running for a short time from t = 0 to t = 20 (top row) and long time from t = 0 to t = 200 (bottom row).

For this particular example, we will use a candidate function library consisting of high order polynomials of the states, up to order 5, and assume that this library is dense enough to capture the dynamics of the Lorenz system in a sparse representation.

$$\boldsymbol{\Theta}(\mathbf{X}) = [\quad \mathbf{1} \quad \mathbf{X} \quad \mathbf{X}^{p_2} \quad \ldots \quad \mathbf{X}^{p_5} \quad] \tag{14}$$

With the denoised states data, a numerical matrix of $\boldsymbol{\Theta}(\mathbf{X})$ can be computed. Also, the state derivative, $\dot{\mathbf{X}}$, can be estimated from the denoised states data using central differences. The calculated $\boldsymbol{\Theta}(\mathbf{X})$ and $\dot{\mathbf{X}}$ are substituted into Algorithm 1. The STLS algorithm can be executed for different threshold parameter λ, which determines the final sparsity of the coefficient matrix Ξ. The number of non-zero terms can be plotted against λ which reveals an L-curve as seen in Figure 4. The threshold parameter is chosen at the Pareto front, which balances between accuracy and complexity. Choosing to threshold at the Pareto front also avoids over-fitting by eliminating the non-important coefficients.

Looking at Figure 4 the pareto front is seen to be at $\lambda = 10^{-1}$ whereby the number of terms plateau as we increase the threshold parameter further. This process is repeated for different signal-to-noise ratios on the states and the coefficients of the candidate functions are tabulated (Tables 1, 2, 3) and placed in a bar chart (Figure 5) below for ease of comparison.

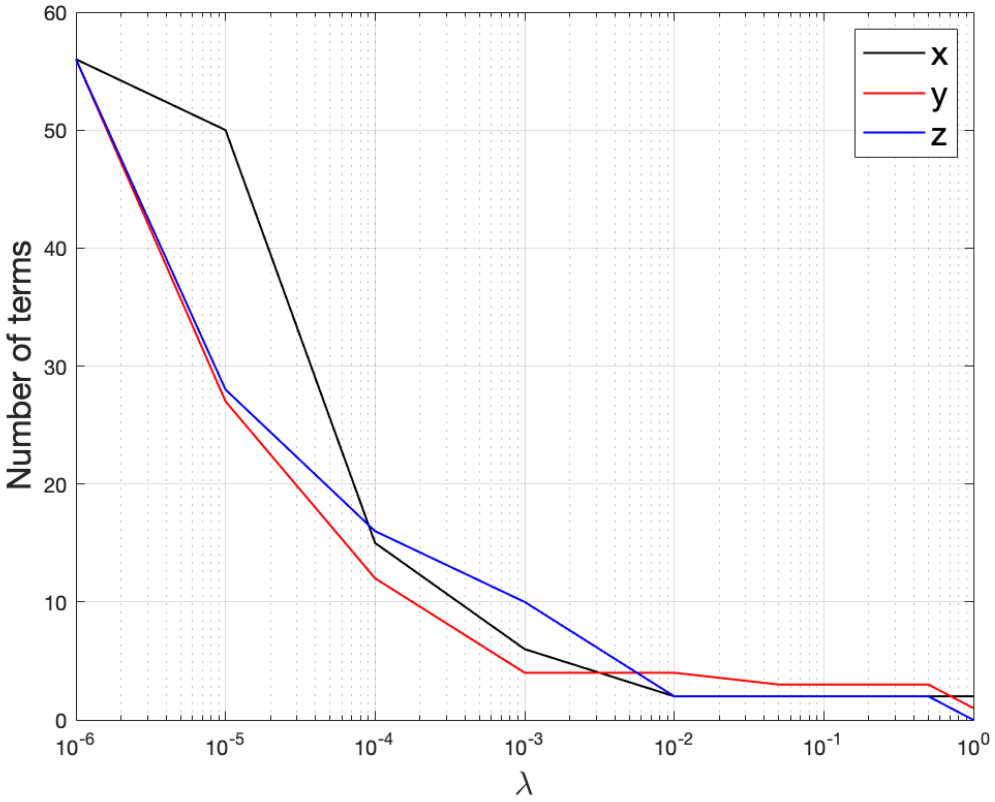


Figure 4: L-curve of the Lorenz system extracted with clean data, the Pareto front was chosen to be $\lambda = 10^{-1}$

Table 1: Identified coefficients of the Lorenz system for various noise levels for state x.

		Candidate Functions						
		1	x	y	z	xy	xz	yz
	Exact		-10	10				
	Clean		-9.98	9.98				
snr [dB]	50		-9.98	9.98				
	40		-9.96	9.96				
	30		-9.72	9.92				
	25		-9.88	9.89				

Table 2: Identified coefficients of the Lorenz system for various noise levels for state y.

		Candidate Functions						
		1	x	y	z	xy	xz	yz
	Exact		28	-1			-1	
	Clean		27.8	-0.96			-0.99	
snr [dB]	50		27.79	-0.96			-0.99	
	40		27.68	-0.91			-0.98	
	30		27.85	-0.9			-0.99	
	25	0.56	27.71	-0.95			-0.99	

Table 3: Identified coefficients of the Lorenz system for various noise levels for state z.

		Candidate Functions						
		1	x	y	z	xy	xz	yz
	Exact				-8/3	1		
	Clean				-2.66	1		
snr [dB]	50				-2.66	1		
	40				-2.66	1		
	30	0.73			-2.65	0.99		
	25	-0.79			-2.56	0.98		

Tables 1, 2, 3 show that the coefficients are well represented for signal-to-noise ratios of 40 and above. Small terms begin appearing at a signal-to-noise ratio of 30 in the z state. The candidate function that makes up each PDE is shown in green highlighted boxes and the red highlighted boxes are the unwanted terms. All empty boxes are zeros.

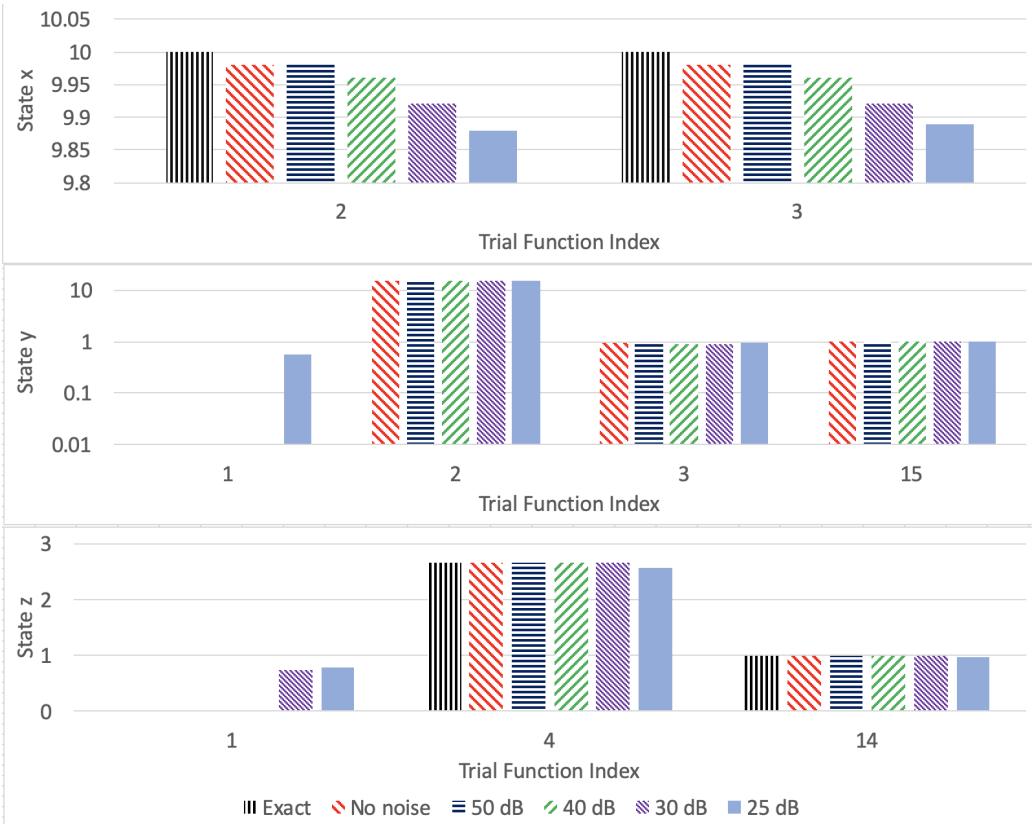
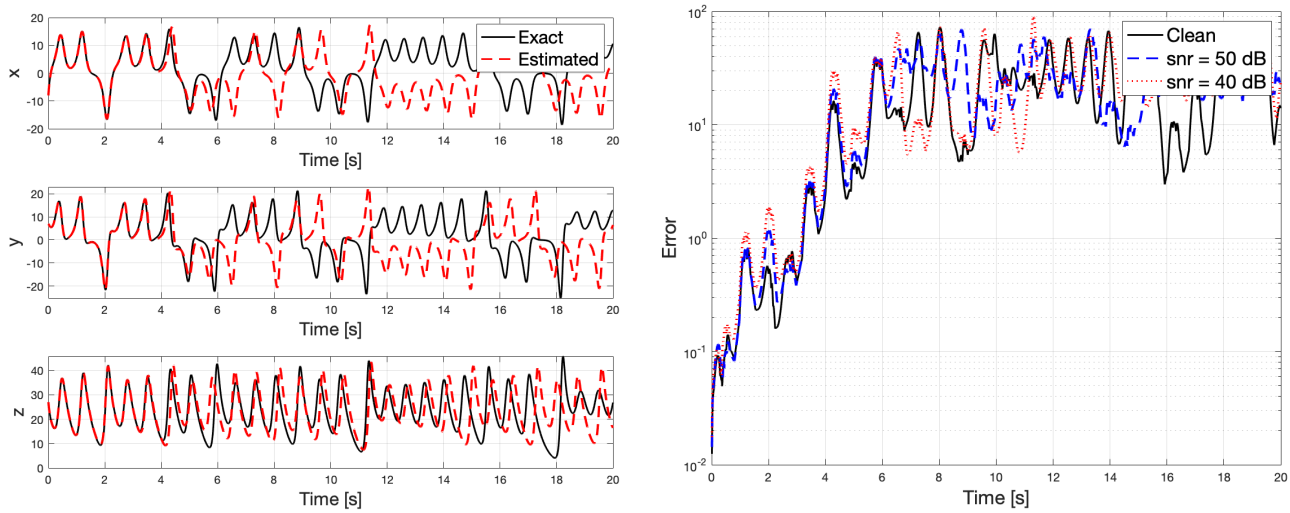


Figure 5: Bar chart containing the absolute value of coefficients for each state at various signal-to-noise ratios for the Lorenz system.

The trial function index in Figure 5 corresponds to the trial functions seen in Table 4 in Appendix A. It can be seen that as the signal-to-noise ratio decrease, the identified coefficients diverges from the exact values and a snr of 30 dB and below, small unwanted constant terms appear in the final identified system. At snr = 40 dB, the extracted coefficients are used as the coefficients of the terms in the Lorenz PDE and the identified system is subsequently solved using the MatLab ODE45 solver, with the same initial condition as the exact trajectory. The solution gives the trajectory of the states as seen in Figure 6a.



(a) Dynamo view

(b) Error

Figure 6: Trajectories of the exact and calculated Lorenz system at snr = 40 dB in dynamo view (a). Absolute difference between the identified and exact trajectories against time at various snr (b).

Since the Lorenz system is ill-conditioned and has a positive Lyapunov exponent, any small deviations in the coefficients caused by numerical rounding would result in the trajectory of the identified system to diverge from the actual trajectory [21]. However, the estimated trajectory follows the exact trajectory for a short time as seen in Figure 6a. Furthermore, at the different noise levels, the error in the trajectory is initially small and increases to a clustered level as seen in Figure 6b.

The Lorenz example has showcased how the STLS algorithm works and how it behaves at different signal-to-noise ratios. In the next section the STLS algorithm will be used on a low dimensional shear flow model, in addition to a state data collection step which forms the preSTLS method seen in Algorithm 2.

3 Sinusoidal Shear Flow

The low dimensional shear flow model example is for the case of sinusoidal shear flow. The evolution equation for this example arises from the Navier-stokes equation with a few assumptions on the form of certain terms. The geometry of this problem can be seen in Figure 7. The characteristic velocity U_0 is the laminar velocity arising from sinusoidal force at y = d/4. Length terms are non-dimensionalized in units of $d/2$, velocities are in units of U_0, time in units of $(\mathrm{d}/2)/U_0$ and pressure in units of $U_0^2\rho$, where ρ is the fluid density [2, 3]. This give rise to the evolution equations

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \nabla p + \frac{1}{Re}\nabla^2\mathbf{u} + \mathbf{F}(y) \tag{15}$$

where the Reynolds number Re is defined as

$$Re = \frac{U_0 d}{2\nu} \tag{16}$$

where ν is the kinematic viscosity of the flow and the fluid is assumed to be incompressible which yields $\nabla\cdot\mathbf{u} = 0$.

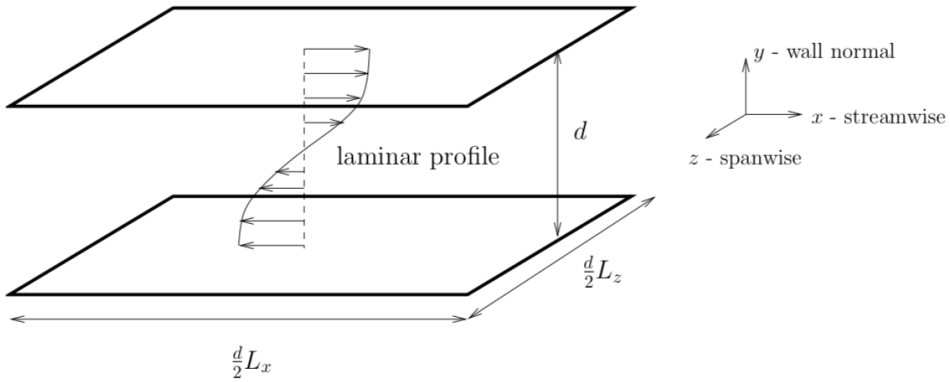


Figure 7: Set-up of the sinusoidal shear flow problem [3].

The walls at $y = \pm d/2$ are assumed to be slip-free

$$v|_{y=\pm d/2} = 0, \quad \frac{\partial u}{\partial y}|_{y=\pm d/2} = \frac{\partial w}{\partial y}|_{y=\pm d/2} = 0 \tag{17}$$

where u, v, w are the velocities in the x, y, z directions respectively. The flow is assumed to be periodic in the x and z directions. The nondimensional volume force is taken to be

$$\mathbf{F}(y) = \frac{\sqrt{2}\pi^2}{4Re} sin(\pi y/2)\hat{\mathbf{e}}_x \tag{18}$$

which results in the laminar profile

$$\mathbf{U}(y) = \sqrt{2} sin(\pi y/2)\hat{\mathbf{e}}_x \tag{19}$$

The length parameter d is taken to be 2 for simplification of the equations. The domain dimensions are therefore $0 \le x \le L_x$, $-1 \le y \le 1$ and $0 \le z \le L_z$. For convenience, we let $\alpha = 2\pi/L_x$, $\beta = \pi/2$ and $\gamma = 2\pi/L_z$.

3.1 Low Dimensional Shear Flow Model

A low-dimensional model was developed for sinusoidal shear flow by Moehlis et al [2] and it is an extension of the original work [10]. This model is based on Fourier modes and is used to describe sinusoidal shear flow whereby fluid between parallel slip-free walls experiences sinusoidal body force. The model contains nine modes, most of which have a direct hydro-dynamical interpretation. The model consists of 9 modes and for completeness all equations are listed below. The coordinates system and domain dimensions are defined in Figure 7. The modes are defined as

$$\text{Basic profile:} \qquad \mathbf{u_1} = \begin{pmatrix} \sqrt{2}sin(\pi y/2) \\ 0 \\ 0 \end{pmatrix} \tag{20}$$

$$\text{Streak mode:} \qquad \mathbf{u_2} = \begin{pmatrix} \frac{4}{\sqrt{3}}cos^2(\pi y/2)cos(\gamma z) \\ 0 \\ 0 \end{pmatrix} \tag{21}$$

$$\text{Downstream vortex:} \qquad \mathbf{u_3} = \frac{2}{\sqrt{4\gamma^2+\pi^2}} \begin{pmatrix} 0 \\ 2\gamma cos(\pi y/2)cos(\gamma z) \\ \pi sin(\pi y/2)sin(\gamma z) \end{pmatrix} \tag{22}$$

$$\text{Spanwise flow:} \qquad \mathbf{u_4} = \begin{pmatrix} 0 \\ 0 \\ \frac{4}{\sqrt{3}}cos(\alpha x)cos^2(\pi y/2) \end{pmatrix} \tag{23}$$

$$\text{Spanwise flow:} \qquad \mathbf{u_5} = \begin{pmatrix} 0 \\ 0 \\ 2sin(\alpha x)sin(\pi y/2) \end{pmatrix} \tag{24}$$

$$\text{Normal vortex:} \qquad \mathbf{u_6} = \frac{4\sqrt{2}}{\sqrt{3(\alpha^2+\gamma^2)}} \begin{pmatrix} -\gamma cos(\alpha x)cos^2(\pi y/2)sin(\gamma z) \\ 0 \\ \alpha sin(\alpha x)cos^2(\pi y/2)cos(\gamma z) \end{pmatrix} \tag{25}$$

$$\text{Normal vortex:} \qquad \mathbf{u_7} = \frac{2\sqrt{2}}{\sqrt{(\alpha^2+\gamma^2)}} \begin{pmatrix} \gamma sin(\alpha x)sin(\pi y/2)sin(\gamma z) \\ 0 \\ \alpha cos(\alpha x)sin(\pi y/2)cos(\gamma z) \end{pmatrix} \tag{26}$$

$$\text{3D mode:} \qquad \mathbf{u_8} = \frac{2\sqrt{2}}{\sqrt{(\alpha^2+\gamma^2)(4\alpha^2+4\gamma^2+\pi^2)}} \begin{pmatrix} \pi\alpha sin(\alpha x)sin(\pi y/2)sin(\gamma z) \\ 2(\alpha^2+\gamma^2)cos(\alpha x)cos(\pi y/2)sin(\gamma z) \\ -\pi\gamma cos(\alpha x)sin(\pi y/2)cos(\gamma z) \end{pmatrix} \tag{27}$$

$$\text{Modified basic profile:} \qquad \mathbf{u_9} = \begin{pmatrix} \sqrt{2}sin(3\pi y/2) \\ 0 \\ 0 \end{pmatrix} \tag{28}$$

Equations 20 to 28 are all defined on a 3-dimensional spatial grid. In Section 3.2, the domain is split into a 10x10x10 grid and substituted into the modal equations above. In Section 3.3 we substitute 3 grid points into the modal equations. For the chosen modes, Moehlis found the following amplitude equations through integration of the evolution equation, see [2] for more details. The follow equations

are the ordinary differential equations (ODEs) that we will attempt to reconstruct from velocity data using the preSTLS algorithm.

$$\frac{da_1}{dt} = \frac{\beta^2}{Re}(1-a_1) - \sqrt{3/2}\frac{\beta\gamma}{\kappa_{\alpha\beta\gamma}}a_6a_8 + \sqrt{3/2}\frac{\beta\gamma}{\kappa_{\beta\gamma}}a_2a_3 \tag{29}$$

$$\begin{aligned}\frac{da_2}{dt} = &-(\frac{4\beta^2}{3}+\gamma^2)\frac{a_2}{Re} + \frac{5\sqrt{2}}{3\sqrt{3}}\frac{\gamma^2}{\kappa_{\alpha\gamma}}a_4a_6 - \frac{\gamma^2}{\sqrt{6}\kappa_{\alpha\gamma}}a_5a_7 \\ &- \frac{\alpha\beta\gamma}{\sqrt{6}\kappa_{\alpha\gamma}\kappa_{\alpha\beta\gamma}}a_5a_8 - \sqrt{3/2}\frac{\beta\gamma}{\kappa_{\beta\gamma}}a_1a_3 - \sqrt{3/2}\frac{\beta\gamma}{\kappa_{\beta\gamma}}a_3a_9\end{aligned} \tag{30}$$

$$\begin{aligned}\frac{da_3}{dt} = &-\frac{\beta^2+\gamma^2}{Re}a_3 + \frac{2}{\sqrt{6}}\frac{\alpha\beta\gamma}{\kappa_{\alpha\gamma}\kappa_{\beta\gamma}}(a_4a_7+a_5a_6) \\ &+ \frac{\beta^2(3\alpha^2+\gamma^2)-3\gamma^2(\alpha^2+\gamma^2)}{\sqrt{6}\kappa_{\alpha\gamma}\kappa_{\beta\gamma}\kappa_{\alpha\beta\gamma}}a_4a_8\end{aligned} \tag{31}$$

$$\begin{aligned}\frac{da_4}{dt} = &-\frac{3\alpha^2+4\beta^2}{3Re}a_4 - \frac{\alpha}{\sqrt{6}}a_1a_5 - \frac{10}{3\sqrt{6}}\frac{\alpha^2}{\kappa_{\alpha\gamma}}a_2a_6 \\ &- \sqrt{3/2}\frac{\alpha\beta\gamma}{\kappa_{\alpha\gamma}\kappa_{\beta\gamma}}a_3a_7 - \sqrt{3/2}\frac{\alpha^2\beta^2}{\kappa_{\alpha\gamma}\kappa_{\beta\gamma}\kappa_{\alpha\beta\gamma}}a_3a_8 - \frac{\alpha}{\sqrt{6}}a_5a_9\end{aligned} \tag{32}$$

$$\begin{aligned}\frac{da_5}{dt} = &-\frac{\alpha^2+\beta^2}{Re}a_5 + \frac{\alpha}{\sqrt{6}}a_1a_4 + \frac{\alpha^2}{\sqrt{6}\kappa_{\alpha\gamma}}a_2a_7 \\ &- \frac{\alpha\beta\gamma}{\sqrt{6}\kappa_{\alpha\gamma}\kappa_{\alpha\beta\gamma}}a_2a_8 + \frac{\alpha}{\sqrt{6}}a_4a_9 + \frac{2}{\sqrt{6}}\frac{\alpha\beta\gamma}{\kappa_{\alpha\gamma}\kappa_{\beta\gamma}}a_3a_6\end{aligned} \tag{33}$$

$$\begin{aligned}\frac{da_6}{dt} = &-\frac{3\alpha^2+4\beta^2+3\gamma^2}{3Re}a_6 + \frac{\alpha}{\sqrt{6}}a_1a_7 + \sqrt{3/2}\frac{\beta\gamma}{\kappa_{\alpha\beta\gamma}}a_1a_8 \\ &+ \frac{10}{3\sqrt{6}}\frac{\alpha^2-\gamma^2}{\kappa_{\alpha\gamma}}a_2a_4 - 2\sqrt{2/3}\frac{\alpha\beta\gamma}{\kappa_{\alpha\gamma}\kappa_{\beta\gamma}}a_3a_5 + \frac{\alpha}{\sqrt{6}}a_7a_9 \\ &\sqrt{3/2}\frac{\beta\gamma}{\kappa_{\alpha\beta\gamma}}a_8a_9\end{aligned} \tag{34}$$

$$\begin{aligned}\frac{da_7}{dt} = &-\frac{\alpha^2+\beta^2+\gamma^2}{Re}a_7 - \frac{\alpha}{\sqrt{6}}(a_1a_6+a_6a_9) \\ &+ \frac{1}{\sqrt{6}}\frac{\gamma^2-\alpha^2}{\kappa_{\alpha\gamma}}a_2a_5 + \frac{1}{\sqrt{6}}\frac{\alpha\beta\gamma}{\kappa_{\alpha\gamma}\kappa_{\beta\gamma}}a_3a_4\end{aligned} \tag{35}$$

$$\begin{aligned}\frac{da_8}{dt} = &-\frac{\alpha^2+\beta^2+\gamma^2}{Re}a_8 + \frac{2}{\sqrt{6}}\frac{\alpha\beta\gamma}{\kappa_{\alpha\gamma}\kappa_{\alpha\beta\gamma}}a_2a_5 \\ &+ \frac{\gamma^2(3\alpha^2-\beta^2+3\gamma^2)}{\sqrt{6}\kappa_{\alpha\gamma}\kappa_{\beta\gamma}\kappa_{\alpha\beta\gamma}}a_3a_4\end{aligned} \tag{36}$$

$$\frac{da_9}{dt} = -\frac{9\beta^2}{Re}a_9 + \sqrt{3/2}\frac{\beta\gamma}{\kappa_{\beta\gamma}}a_2a_3 - \sqrt{3/2}\frac{\beta\gamma}{\kappa_{\alpha\beta\gamma}}a_6a_8 \tag{37}$$

Where

$$\kappa_{\alpha\gamma} = \sqrt{\alpha^2+\gamma^2} \quad \kappa_{\beta\gamma} = \sqrt{\beta^2+\gamma^2} \quad \kappa_{\alpha\beta\gamma} = \sqrt{\alpha^2+\beta^2+\gamma^2} \tag{38}$$

The ODEs given by Equations 29 to 37 are solved using Matlab's ODE45 solver to give $\mathbf{a}(t)$ and the flow velocity at a given point in space and time is given by:

$$\mathbf{u}(\mathbf{x},t) = \sum_m a_m(t)\mathbf{u}_m(\mathbf{x}) \tag{39}$$

Where spatial operators $\mathbf{u}_m(\mathbf{x})$ are the modes given by Equations 20 to 28. This results in the following flow contour.

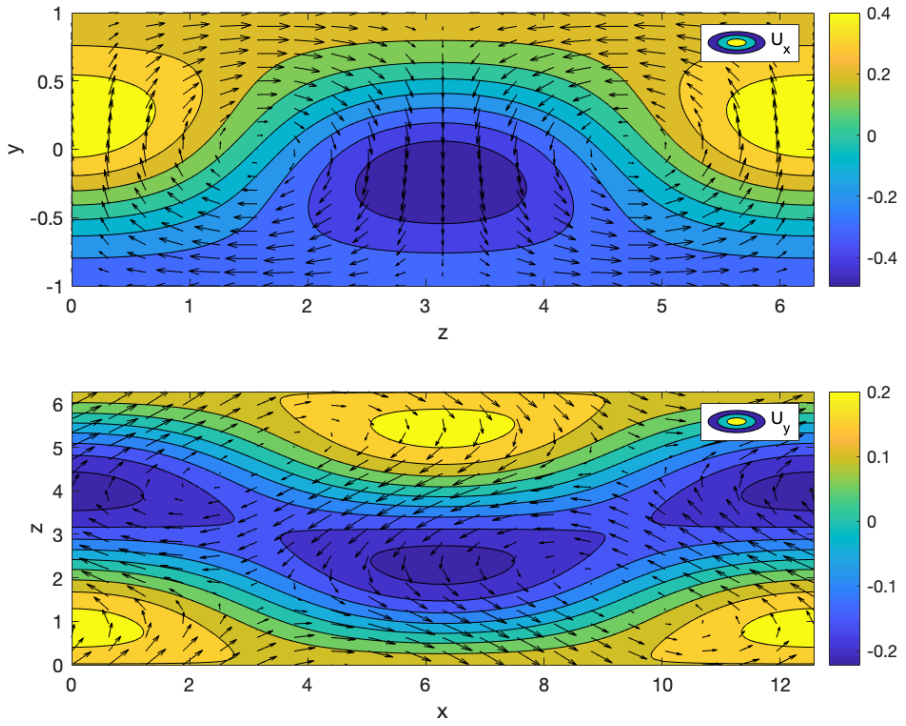


Figure 8: Flow contour at t = 1s, for Re = 300, $L_x = 4\pi$ and $L_y = 2\pi$.

To replicate real life application of the STLS methodology, $\mathbf{u}(\mathbf{x},t)$ that was found in the previous calculations is used as the raw data. We begin by adding white Gaussian noise to the velocity data $\mathbf{u}(\mathbf{x},t)$ followed by denoising of $\mathbf{u}(\mathbf{x},t)$ using the **lowess** method. The reason for using the **lowess** denoising is discussed in Section 2.1.2. Note that in this example, the denoising step occurs prior to the extraction of the states, whereas the Lorenz example we assume that we collect directly the noisy states data. Next, spatial coordinates inside the domain are substituted into Equations 20 to 28, and the states $a_m(t)$ are found by using least squares regression on Equation 39 at each time step. The estimated $a_m(t)$ are then used in the STLS algorithm in order to find the coefficients of the ODEs. This is highlighted in Algorithm 2. In this example, a sampling time step of $\Delta t = 0.05s$ is used, and as discussed in Section 2.1.2, **lowess** works better than the wavelet denoising.

Algorithm 2: Preliminary state extraction from velocity data and STLS

function$[\xi]$ = **preSTLS**($\mathbf{u}(\mathbf{x},t)$, $\mathbf{u}(\mathbf{x})$, λ)
for $(t_i = 0,\ t_i < t_f,\ t_i++)$ **do**
  $\mathbf{a}(t_i) = \mathbf{u}(\mathbf{x}) \setminus \mathbf{u}(\mathbf{x}, t_i)$; % *find* $\boldsymbol{a}(t)$ *using least squares at each time step*
Calculate $\boldsymbol{\Theta}(\mathbf{a})$ and $\dot{\mathbf{a}}(t)$;
ξ = **STLS**($\boldsymbol{\Theta}(\mathbf{a})$, $\dot{\mathbf{a}}(t)$, λ)

Equation 39 can be written in matrix form as follows:

$$\begin{bmatrix} u(\mathbf{x},t) \\ v(\mathbf{x},t) \\ w(\mathbf{x},t) \end{bmatrix} = \begin{bmatrix} u_{11}(\mathbf{x}) & u_{21}(\mathbf{x}) & \cdots & u_{91}(\mathbf{x}) \\ u_{12}(\mathbf{x}) & u_{22}(\mathbf{x}) & \cdots & u_{92}(\mathbf{x}) \\ u_{13}(\mathbf{x}) & u_{23}(\mathbf{x}) & \cdots & u_{93}(\mathbf{x}) \end{bmatrix} \begin{bmatrix} a_1(t) \\ a_2(t) \\ \vdots \\ a_9(t) \end{bmatrix} \tag{40}$$

Where $\mathbf{u}(\mathbf{x},t) = [u(\mathbf{x},t) \quad v(\mathbf{x},t) \quad w(\mathbf{x},t)]^T$ are the velocities in the x, y, z directions respectively. In addition, the velocities are collected at p different sensor points, and are grouped together in their different directional components, i.e.

$$u(\mathbf{x},t) = [u(x_1,y_1,z_1,t) \quad u(x_2,y_2,z_2,t) \quad ... \quad u(x_p,y_p,z_p,t)]^T \tag{41a}$$
$$v(\mathbf{x},t) = [v(x_1,y_1,z_1,t) \quad v(x_2,y_2,z_2,t) \quad ... \quad v(x_p,y_p,z_p,t)]^T \tag{41b}$$
$$w(\mathbf{x},t) = [w(x_1,y_1,z_1,t) \quad w(x_2,y_2,z_2,t) \quad ... \quad w(x_p,y_p,z_p,t)]^T \tag{41c}$$

This makes the LHS of Equation 40 a $3p \times 1$ vector. Furthermore, the $\mathbf{u}(\mathbf{x})$ matrix on the RHS consists of values from the spatial distributions given by Equations 20 to 28, where $u_{i,j}(\mathbf{x})$ is a vector that contains the fixed-in-time values for the j^{th} component of the i^{th} state at various points in space, for instance

$$\begin{aligned} u_{11}(\mathbf{x}) &= [u_{11}(x_1,y_1,z_1) \quad u_{11}(x_2,y_2,z_2) \quad ... \quad u_{11}(x_p,y_p,z_p)]^T \\ &= [\sqrt{2}sin(\pi y_1/2) \quad \sqrt{2}sin(\pi y_2/2) \quad ... \quad \sqrt{2}sin(\pi y_p/2)]^T \end{aligned} \tag{42}$$

This makes the $\mathbf{u}(\mathbf{x})$ matrix with dimensions $3p \times 9$. Using velocity data from fewer than 3 points will lead to an under-determined problem, which in general has infinitely many solutions, or no solutions if the equations are inconsistent. In this case, least squares can be implemented to ensure a solution exists and we can choose from the many possible solutions. Constraints are usually added to the under-determined least squares problem to ensure a unique solution [22]. Selecting more than 3 points will lead to an over-determined system, which is solvable using least squares. The critical case of 3 points will lead to a problem with same number of unknowns as equations and if possible can be solved using the inverse of the matrix. In Section 3.2, we look at how Algorithm 2 performs using an over-determined system with 10x10x10 sensor points evenly distributed across the domain. In Section 3.3, we utilise the critical case of 3 sensor points, and discuss the idea of using 2 sets of 3 sensor points to reconstruct all states accurately. Unless stated otherwise, all examples of the low-dimensional shear flow in later sections will share the same initial condition

$$\mathbf{a}(0) = [\quad 0.21, \quad 0.23, \quad 0.17, \quad 0.23, \quad 0.17, \quad 0.16, \quad 0.13, \quad 0.15, \quad 0.16 \quad] \tag{43}$$

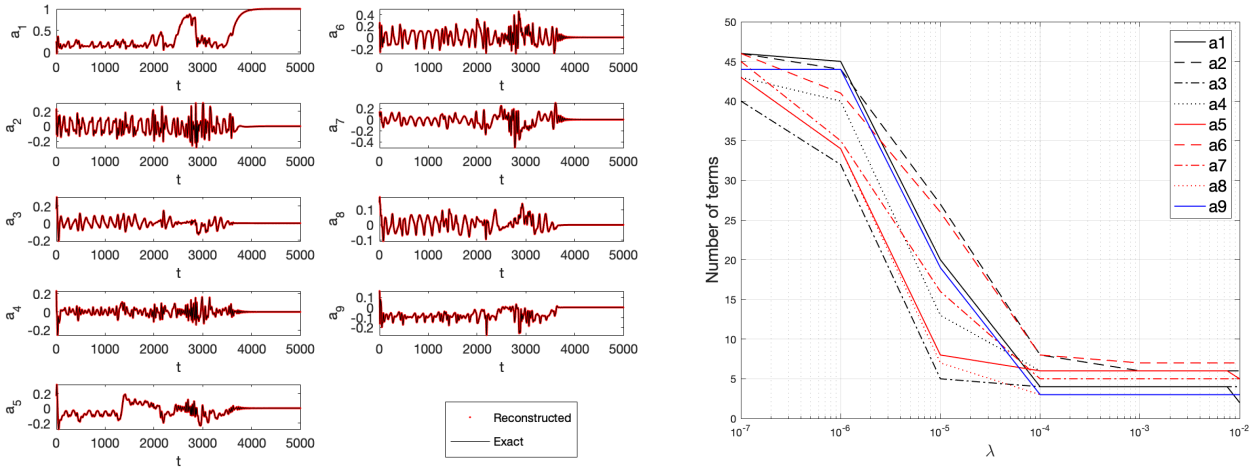
Which was selected to satisfy the energy cap of $\sum_i a_i^2 = 0.3$. The library of candidate function was chosen to consist of a constant and combination of states up to order 2

$$\boldsymbol{\Theta}(\mathbf{A}) = [\quad \mathbf{1} \quad \mathbf{A} \quad \mathbf{A}^{p_2} \quad] \tag{44}$$

Where $\mathbf{A} = [a_1 \quad a_2 \quad ... \quad a_9]$.

3.2 Over-determined States Extraction

We first begin by attempting to find the coefficients of the low-dimensional shear flow model given by Equations 29 to 37, using 10x10x10 sensor points, uniformly distributed, on the domain defined with $L_x = 4\pi$ and $L_z = 2\pi$. This gives us 1000 data extraction points and thus 3000 velocities to infer the states data from. This will require us to solve an over-determined system of equations to find $\mathbf{a}(t)$ in Algorithm 2. Noise with various snr values are added to the velocity data and subsequent denoising was conducted on the velocity data using **lowess**, as discussed in Section 3.1. The denoised velocity data is then inserted into the first part of Algorithm 2 to extract the states data. Figure 9a shows the reconstructed states data (red dots) alongside the original states data (black line).



(a) Extracted States

(b) L-curve

Figure 9: Data extraction from a 10x10x10 grid, solving an over-determined least squares problem to find $\mathbf{a}(t)$. No noise, Re = 300, $L_x = 4\pi$ and $L_z = 2\pi$, initial condition from Equation 43.

The values $a_1 = 1$ and all other states equals zero corresponds to the laminar flow conditions. For this example, the flow transitioned from turbulent to laminar at around t = 4000s. The reconstructed states data, $a_1(t)$ to $a_9(t)$ are substituted into the candidate function library given in Equation 44. Note that in this example we only denoise velocity data, and not the states. State derivatives $\dot{\mathbf{a}}(t)$ are estimated using central difference from our reconstructed states data. Finally, the state derivative $\dot{\mathbf{a}}(t)$ and the library of candidate functions $\boldsymbol{\Theta}(\mathbf{A})$ are substituted into the STLS algorithm. Selecting the appropriate thresholding parameter λ from Figure 9b results in the coefficients seen in the Figure 10. In this case $\lambda = 10^{-3}$ was chosen.

The trial function index corresponds to the trial functions seen in Appendix A. Generally, the extracted coefficients are acceptably accurate down to a snr value of 60 dB. At 50 dB the preSTLS algorithm fails to identify correctly the coefficients. Whereas, the Lorenz system failed for snr values lower than 40 dB. This greater sensitivity to noise for the shear flow model is most probably due to the added noise amplification from the additional step of extracting the states $a_m(t)$ from noisy velocity measurements using least squares.

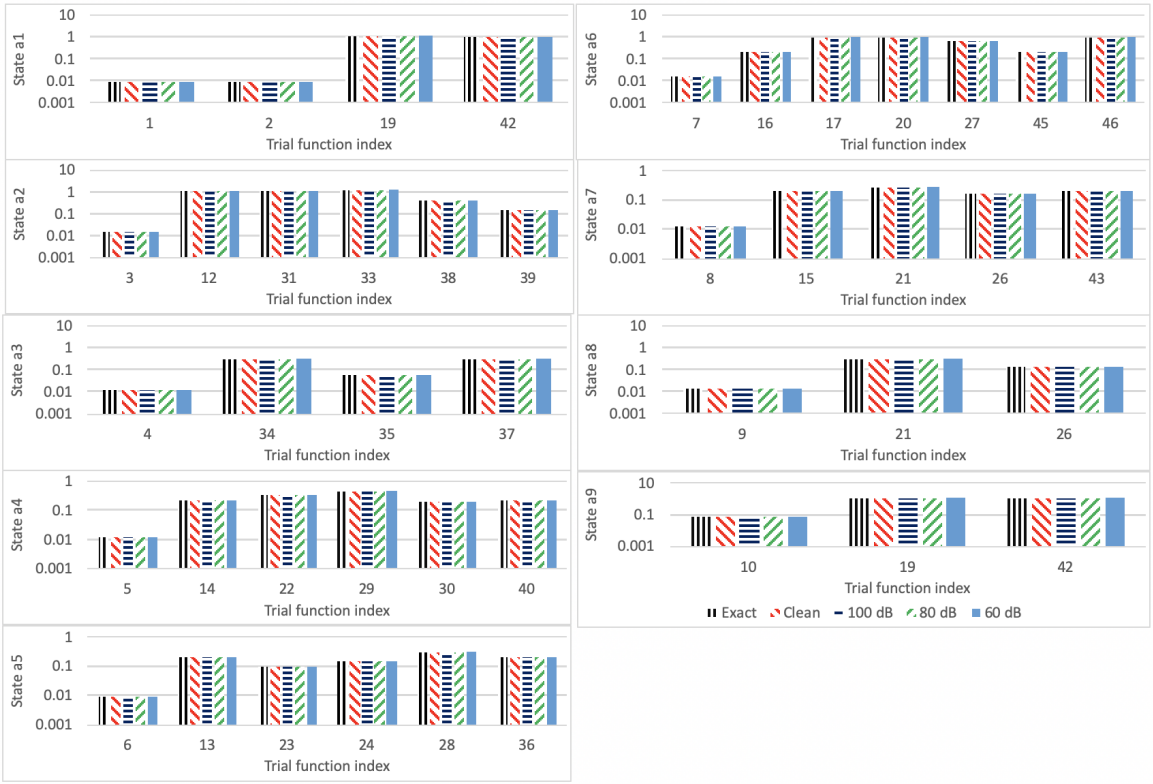


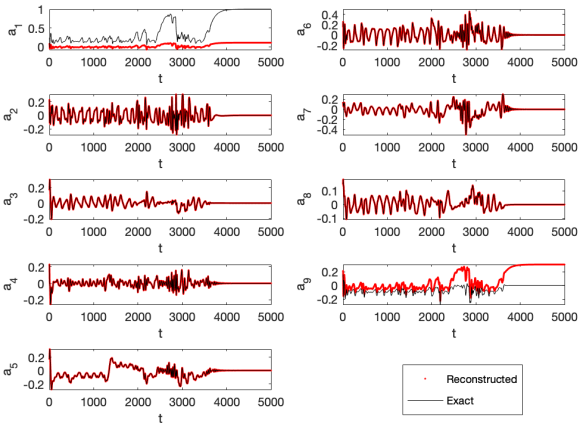
Figure 10: Absolute coefficients of the ODEs for the low-dimensional turbulent shear flow model at various snr values, with velocity data extracted from a 10x10x10 grid on the domain. Re = 300, $L_x = 4\pi$ and $L_z = 2\pi$.

3.3 States Estimation from Fewer Velocity Extraction Points

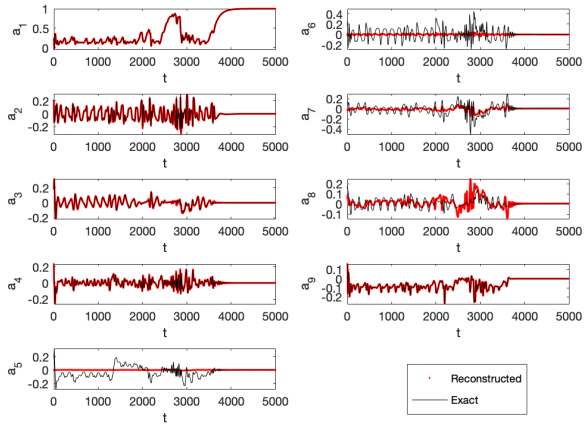
In reality, it may be costly to extract velocity data from a 10x10x10 grid, however it is possible to estimate the states $\mathbf{a}(t)$ using a minimum of 3 grid points, with 3 velocities at each grid point, this gives 9 velocities in which to solve for the 9 states. Using less than grid 3 points will require solving an under-determined least squares problem, as previously discuss might have infinitely many solutions. So to avoid solving under-determined or over-determined systems, 3 extraction points should be used. However, because the fluid flow has 3-dimensional modes, to capture all modes accurately using only 3 points in space is difficult. So two iterations of the data extraction using 3 points are conducted. When using 3 points in the x-z plane, for a fixed y, all states are extracted accurately apart from from a_1 and a_9, as seen in Figure 11a. Whereas using 3 points on a y-line for a fixed x and z returned much better a_1 and a_9, as seen in Figure 11b.

So, combining the best of both we can capture fairly accurately all states using only 6 points, or 5 points if they share a common point. For simplicity, the coordinates of the points selected on the x-z plane are as follows the [(0, 0.01, 0), (0.01, 0.01, 0.01), (0.02, 0.01, 0.02)] and the y-line points are [(0.01, 0, 0.01), (0.01, 0.01, 0.01), (0.01, 0.02, 0.01)]. Both extractions share the point (0.01, 0.01, 0.01), so only 5 points are required.

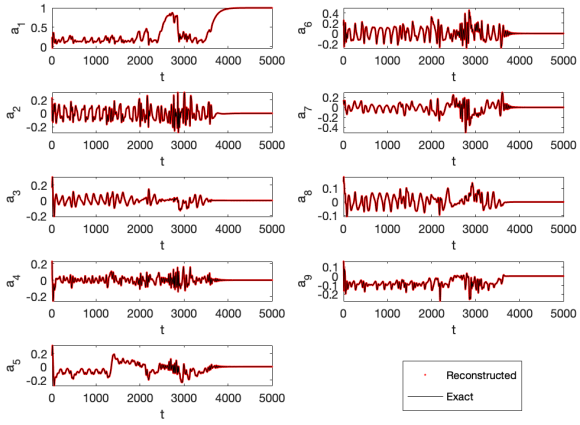
Note that prior to the reconstruction of the states data, the velocity data is treated with the same additive noise and subsequent denosing as seen in Section 3.2, and we do not treat the states with any denoising.



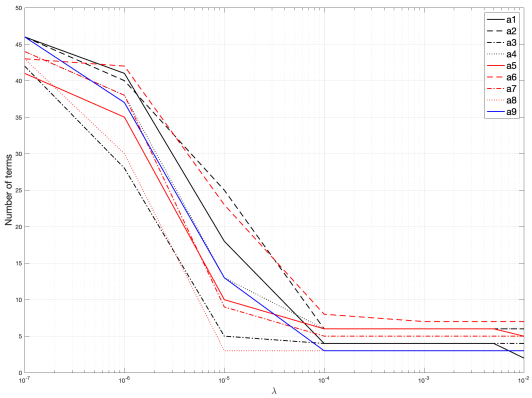
(a) xz-plane



(b) y-line



(c) Combined



(d) L-curve

Figure 11: Data extraction from 3 points on the xz-plane for a fixed y (a) and 3 points on a y-line for fixed x and z (b). The combined best of both used in later calculations (c). L-curve from using combined states(d) No noise, Re = 300, $L_x = 4\pi$ and $L_z = 2\pi$, initial condition from Equation 43.

The resulting combined reconstructed states data seen in Figure 11c, are substituted into the candidate function library $\mathbf{\Theta}(\mathbf{A})$ in Equation 44. The state derivatives $\dot{\mathbf{a}}_m(t)$ are estimated using central difference. Both $\mathbf{\Theta}(\mathbf{A})$ and $\dot{\mathbf{a}}_m(t)$ are substituted into the STLS algorithm. The L-curve of this example is seen in Figure 11d and a thresholding parameter of $\lambda = 10^{-3}$ was chosen as the Pareto front. Using only 5 grid points in the velocity extraction, results in successful replication of the trial function coefficients as seen in Figure 12. However, this method fails at a noticeably higher snr value of 150 dB (not shown as the bar plot becomes too crowded and difficult to read). This error is due to the failure of extracting the states using least squares.

Therefore, it is seen that the coefficients of the underlying ODEs were captured to an acceptable accuracy from using only 5 sensor points for relatively low noise levels. We can then feed these coefficients back into the ODE and solving for the states again, starting with the same initial conditions defined in Equation 43. The identified and exact state trajectories can be seen in Figure 13a. The reconstructed states follow the original states for a short period of time and begin to deviate at t = 800s. Both trajectories exhibit finite lifetimes of the turbulent regime. The finite lifetime of the trajectories suggests that the turbulent state is not an attractor, but rather a chaotic saddle, whereby there is eventual decay to the transient state [2].

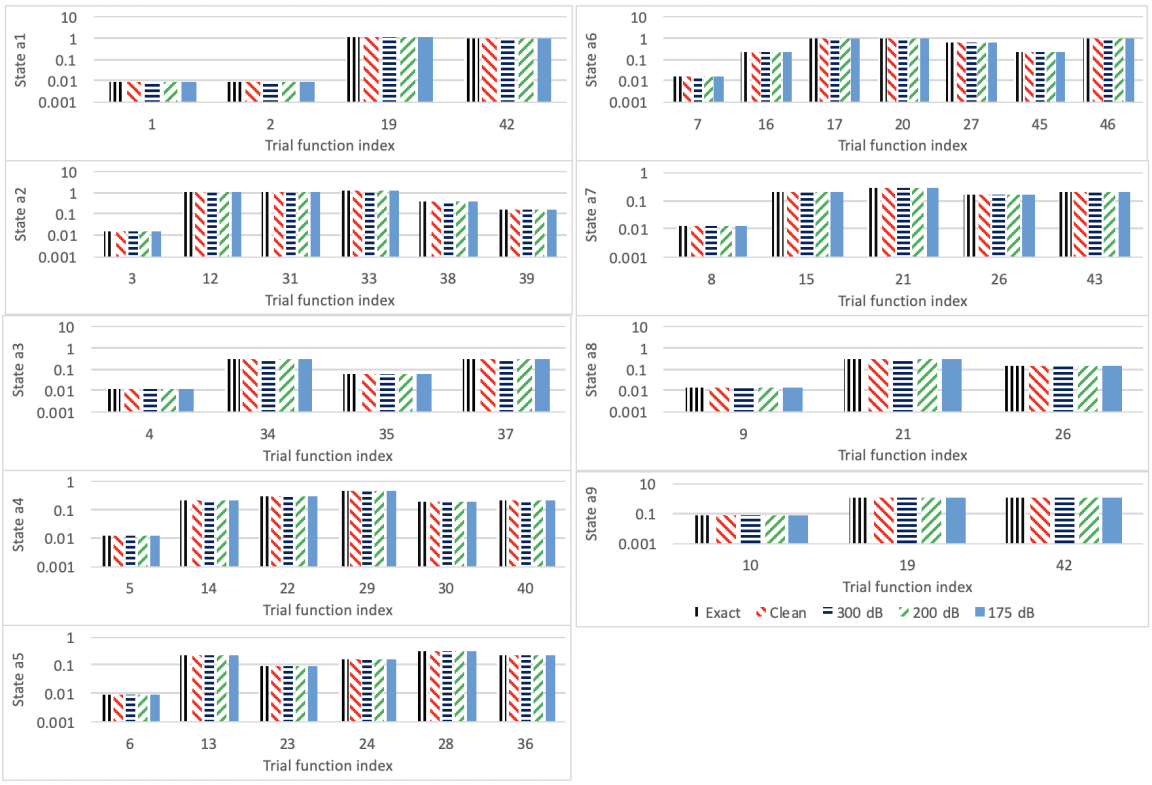
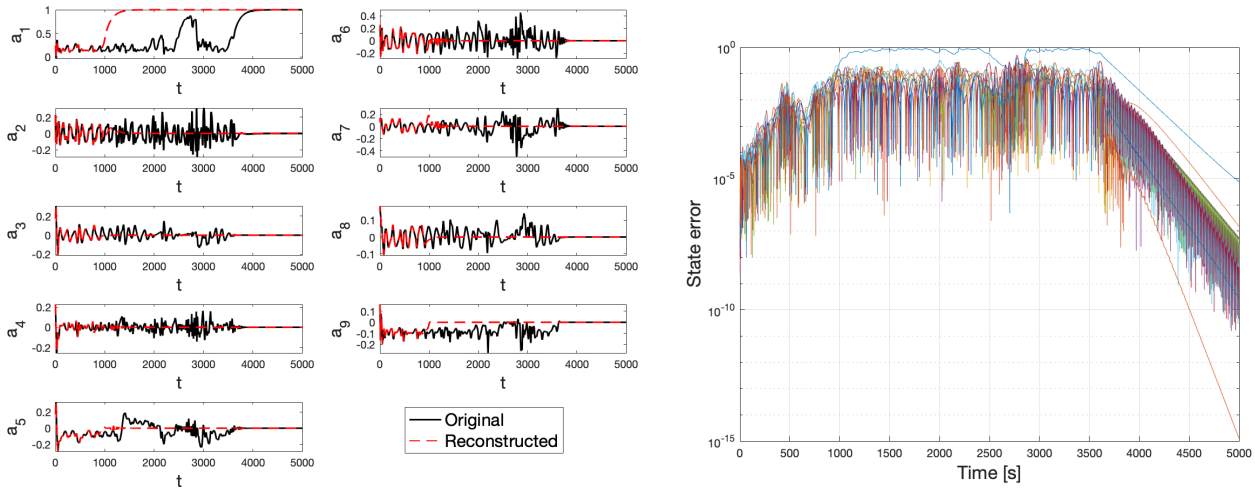


Figure 12: Absolute coefficients of the ODEs for the turbulent shear flow model at various snr values, with velocity data extracted from 5 grid points in the domain. Re = 300, $L_x = 4\pi$ and $L_z = 2\pi$.

The difference is better seen in Figure 13b, whereby the error is initially small and increases until it saturates at t = 800s. This is justified as the shear flow model has a positive Lyapunov exponent, whereby small differences in the model coefficients or initial conditions will grow exponentially until saturation [21]. Eventually both settles back to the stable laminar regime and a decline in error is seen from t = 3500s.



(a) Trajectories

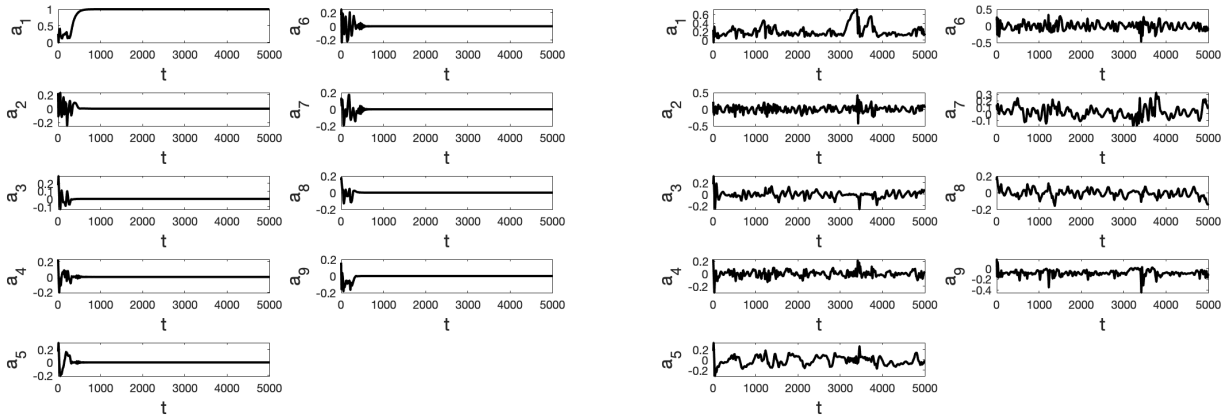
(b) Error

Figure 13: Reconstructed states compared to the original states data (a). Error of the reconstructed states compared to the original states data with time (b). No noise, Re = 300, $L_x = 4\pi$ and $L_z = 2\pi$. Using 5 sensor points.

4 Parametric Analysis

4.1 Effect of Reynolds Number

The flow behaviour heavily depends on the Reynolds number, more specifically the time in which transition occurs. Generally, the flow remains turbulent for longer at higher Reynolds number. However, there is a complex fractal relationship when the initial perturbation energy is also varied. This is discussed in later sections. All subsequent estimations of $\mathbf{a}(t)$ are from using the 5-point velocity extraction described as in the previous section.



(a) Re = 200 (b) Re = 500

Figure 14: Variation of states with time at different Reynolds number. No noise, $L_x = 4\pi$ and $L_z = 2\pi$, initial condition from Equation 43.

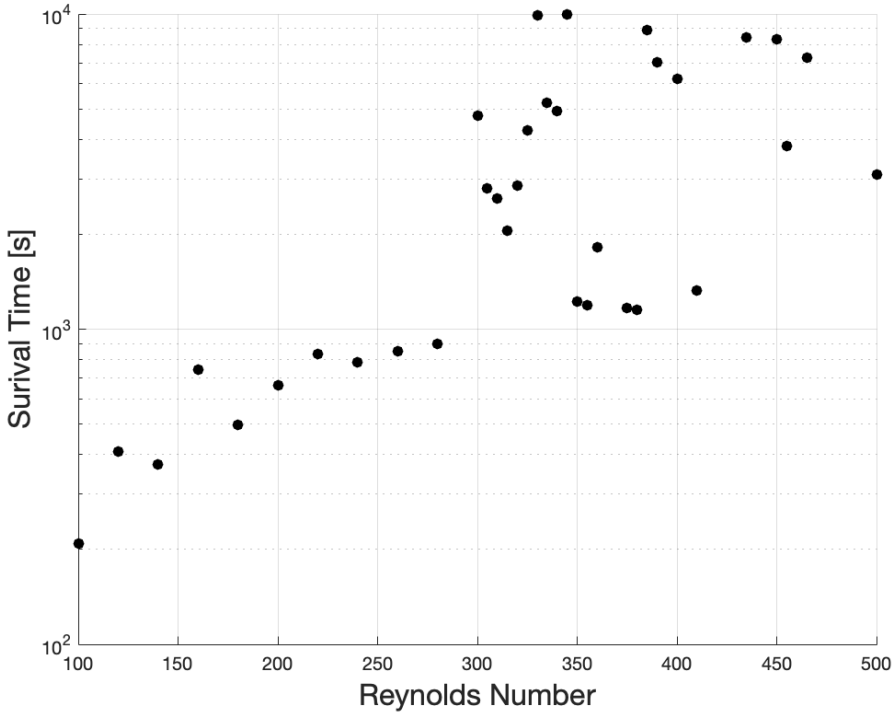


Figure 15: Lifetime of the turbulent regime against Reynolds number. No noise, $L_x = 4\pi$ and $L_z = 2\pi$.

As discussed before, $a_1 = 1$ and $a_2, ..., a_9 = 0$ represents the laminar case. For convenience, we will define turbulent lifetime or survival time as the time from $t = 0$ to when a_1 reaches a value 0.99. Looking at Figure 15 we can see that generally as we increase the Reynolds number the lifetime of the turbulent regime increases and above Re = 300, the lifetimes seems to be more scattered. Here the initial energy of the perturbations are fixed and defined in Equation 43. The scattering of the survival time above Re = 300 suggests a more complex underlying relationship with the energy of the initial perturbation. We can subsequently extract the coefficients of the ODEs using STLS at various Reynolds number. Looking at Figure 16 the extracted coefficients are least accurate for Re = 200, and the accuracy improves as we increase the Reynolds number. Though not shown in the graph, the STLS algorithm fails to return the correct coefficients at a Re = 100.

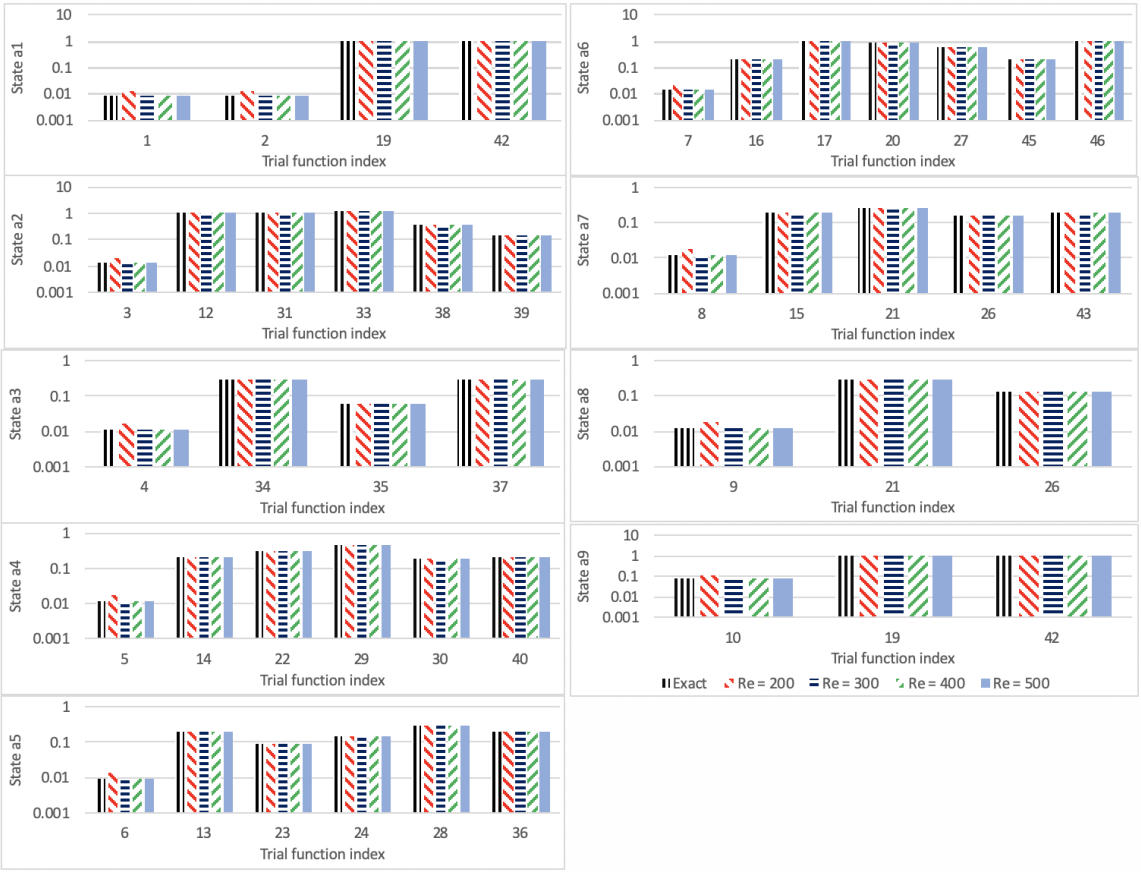


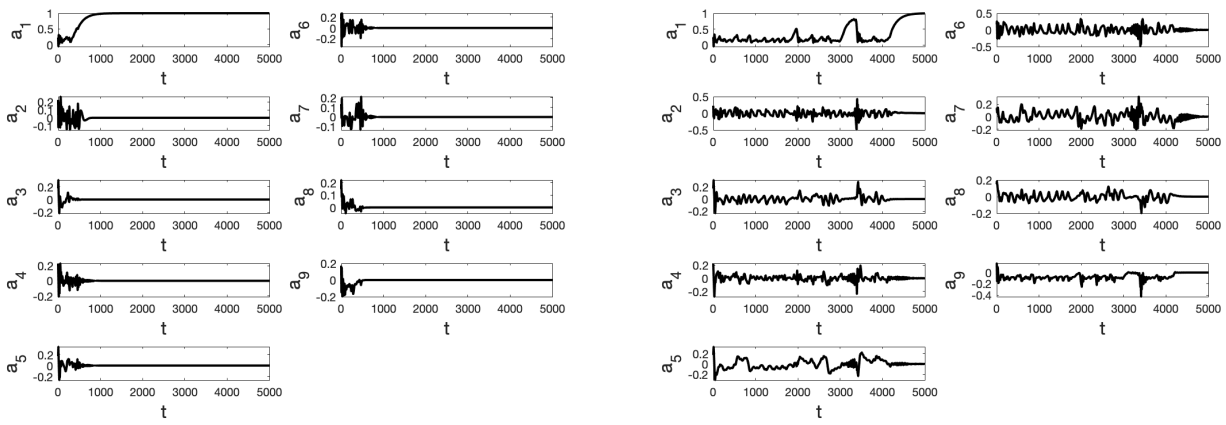
Figure 16: Absolute coefficients of the ODEs for the low-dimensional turbulent shear flow model at various Reynolds number. No noise, $L_x = 4\pi$ and $L_z = 2\pi$.

Note that the relationship between survival time and Reynolds number is intrinsic to the model, and independent to the signal-to-noise ratio that we use. However, the snr value greatly impacts the relationship between survival time and the absolute error between the extracted and exact ceofficients of the ODEs. The map from survival time to error is explored in Section 4.4.

4.2 Effect of Domain Size

The domain size with $L_x = 1.75\pi$ and $L_z = 1.2\pi$ was found to be the minimal flow unit, in which it is the smallest domain that turbulence can numerically be sustained [23]. The larger domain with $L_x = 4\pi$ and $L_z = 2\pi$ was found to be the optimal domain size for plane Couette flow, where finite amplitude solutions exists at the smallest Reynolds number [24]. So these two cases are compared for Re = 300 and fixed initial conditions defined in Equation 43. In comparison, the dynamics of the

larger domain is more violent and less regular than the smaller one, but both share similar vortex and sinusoidal instabilities. For the same initial perturbation energy and Reynolds number the minimal flow unit domain has a shorter turbulent lifetime as compared to the optimal domain size, so one would expect the optimal domain size to return better coefficient results. The optimal domain size will be used for later analysis to compare a wider range of Reynolds number, as we can sustain turbulence at much lower Reynolds numbers.



(a) $L_x = 1.75\pi$ and $L_z = 1.2\pi$ (b) $L_x = 4\pi$ and $L_z = 2\pi$

Figure 17: Variation of states with time with the minimal flow unit domain (a) and optimal domain size (b). No noise, Re = 300, initial condition from Equation 43.

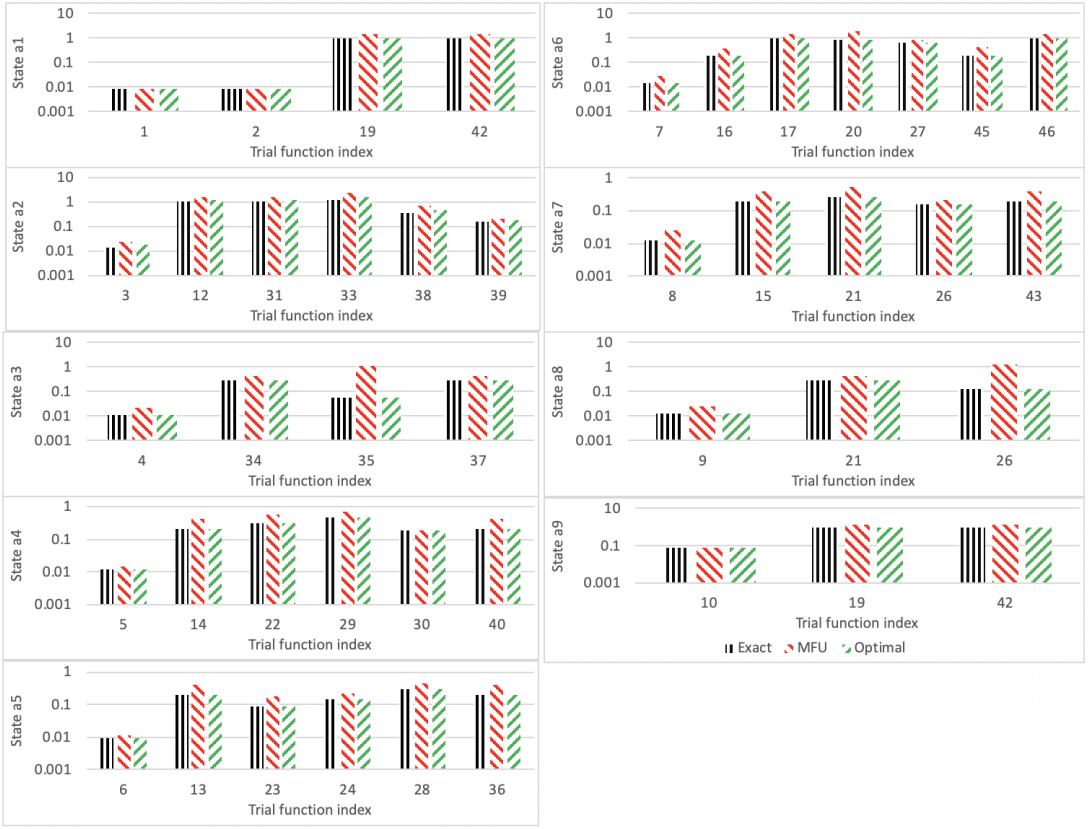


Figure 18: Absolute coefficients of the ODEs for the low-dimensional turbulent shear flow model at minimal flow unit (MFU) and optimal domain sizes. No noise and Re = 300.

4.3 Variation of Initial Perturbation Energy

The simulation of the flow, more specifically the turbulent and laminar interaction is very sensitive to the initial conditions for low Reynolds number. In previous sections, the initial conditions were fixed as given by Equation 43 for comparison purposes. In this section we explore how the energy of the initial condition affects the preSTLS algorithm. Figure 19 shows the fractal dependence of turbulence lifetime on different perturbation energy (E) and Reynolds number (Re). The initial conditions here are $a_1 = a_6 = a_7 = a_8 = a_9 = 0$ and $a_2 = a_3 = a_4 = a_5 = \sqrt{E}/2$, as defined by Moehlis et al [2]. The optimal domain size was used, with $L_x = 4\pi$ and $L_z = 2\pi$ to ensure that lifetimes behaviours are captured at the lower Reynolds numbers. As discussed in Section 4.1, the turbulence lifetime is defined as the first time in which a_1 reaches 0.99.

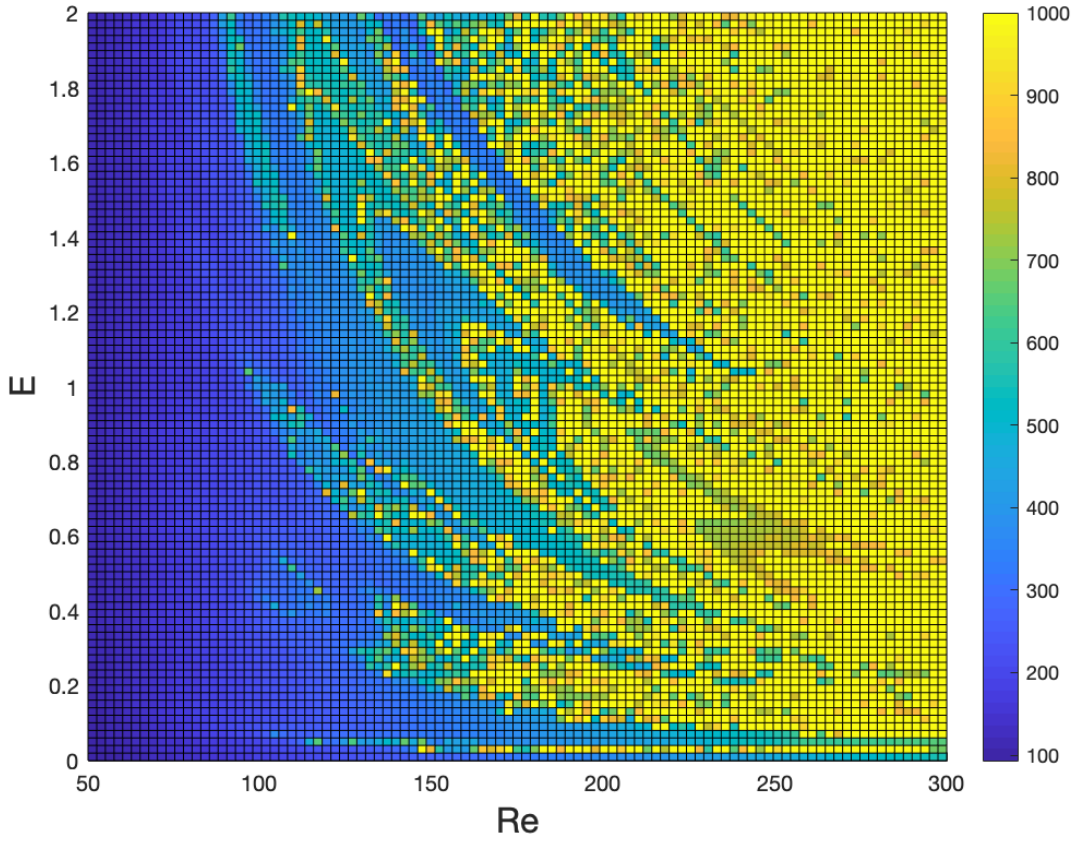


Figure 19: Turbulence lifetime in seconds (color-bar) for various perturbation energy (E) and Reynolds number (Re).

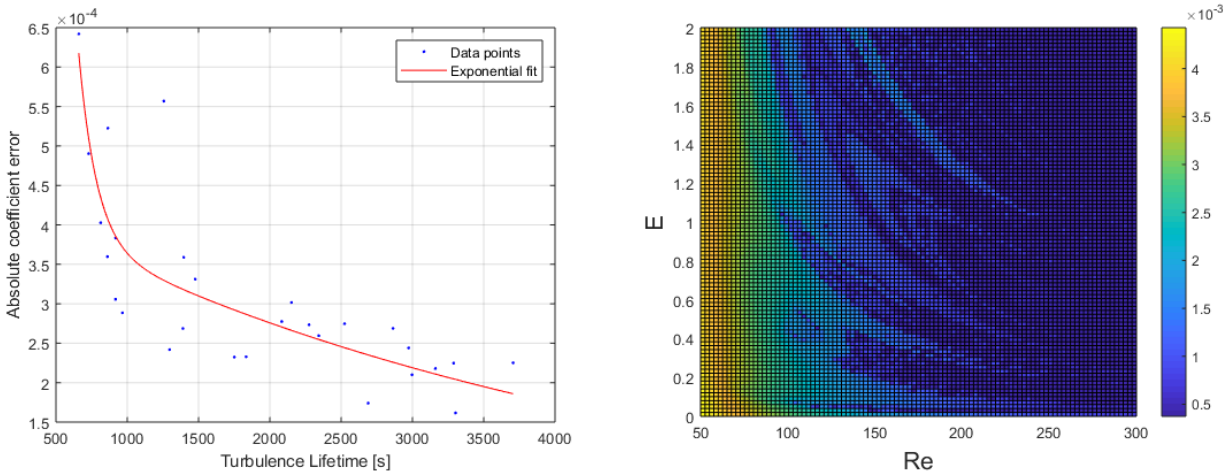
Looking at Figure 19, we can see that for Reynolds numbers below 75, the turbulence lifetime is very short and varying the initial perturbation energy does not apparently change the lifetimes that much. For slightly higher Re of around 100 to 200 the lifetime is very sensitive to variation in initial energy, where fractal behaviours are observed, with flame-like patterns emerging inside one another. For large Re of around 300 the lifetimes are saturated at 1000 seconds. This would suggest that at high Re, turbulence is sustained even for low energy perturbations, which makes some intuitive physical sense.

4.4 Error Analysis

From Sections 4.1 and 4.2 it would seem that the accuracy of the identified coefficients is better for longer lifetimes. This due to the fact the laminar profile ($a_1 = 1$ and others equals zero) provides no relevant information on the modes. As such, the most accurate coefficients are returned when there is a sufficient abundance of turbulent state data. In this section, error is defined as the sum of the absolute difference between the identified and exact coefficients of the ODEs or the L^1 norm of the difference. To achieve variations in survival times, the states data were collected for random initial conditions with an energy cap defined by $\sum_i a_i^2 = 0.3$. All other simulation parameters were fixed, with simulation run time of 4000s, no added noise, optimal domain size and at a fixed Re = 300. Data points where the flow remained turbulent for the entirety of 4000s were neglected.

4.4.1 Using 10x10x10 Velocity Extraction Points

For the case of using an even distribution of 10x10x10 sensor points, as seen in Section 3.2, we can measure the absolute difference between the identified and exact ODE coefficients. This is conducted for different turbulence lifetimes and Figure 20a can be drawn.



(a) Error against lifetimes

(b) Fractal error

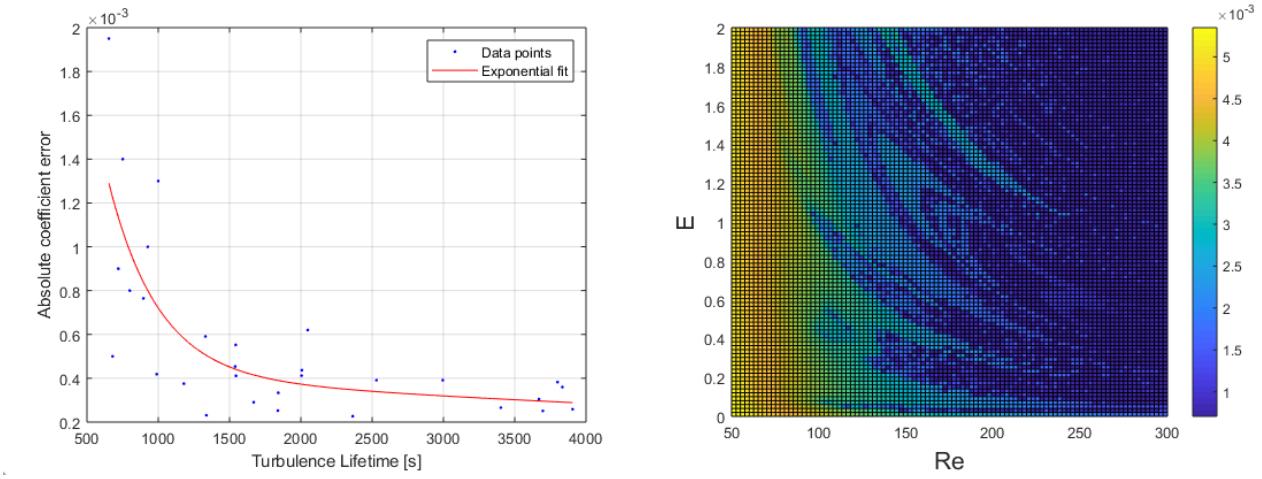
Figure 20: Absolute error between the identified coefficients and the exact coefficients plotted against survival time of the turbulent regime for the shear flow model (a). Absolute error (color-bar) for various perturbation energy (E) and Reynolds number (Re). Using 10x10x10 points extraction with no noise. Exponential fit $y = 0.0065e^{-0.00514x} + 0.00042e^{-0.00021x}$ (a).

Upon further inspection of the relationship between error of the identified states and lifetimes of turbulence, an exponential decay relationship was identified. With the fractal dependence of turbulence lifetime on different perturbation energy and Reynolds number (Figure 19), and an exponentially decaying relationship between error and survival time (Figure 20a), the survival time can be mapped to error to give Figure 20b.

Figure 20b also shows an apparent fractal relationship between the error of the identified ODE coefficients on Re and E, similar to that seen in Figure 19. However, the error exponentially increases as we decrease Re below 75.

4.4.2 Using 5 Velocity Extraction Points

We can repeat the process as in Section 4.4.1 for the 5 sensor points as discussed in Section 3.3. There is also an exponential decay relationship between absolute coefficient error and turbulence lifetimes. However, the errors here are approximately of order 10 larger than that of the 10x10x10 sensor points example.



(a) Error against lifetimes

(b) Fractal error

Figure 21: Absolute error between the identified coefficients and the exact coefficients plotted against survival time of the turbulent regime for the shear flow model (a). Absolute error (color-bar) for various perturbation energy (E) and Reynolds number (Re). Using 5 sensor points with no noise. Exponential fit $y = 0.0065e^{-0.00301x} + 0.00041e^{-0.00007x}$. (b).

The error in the identified coefficients plateaus at survival times around 2000s and above, suggesting that Algorithm 2 returns the best results when at least 50% of the collected velocity is in the turbulent regime. Once again, we can map the turbulent lifetimes in Figure 19 using the errors in Figure 21a to give Figure 21b. Once again, we observe a fractal relationship between the error of the identified ODE coefficients on Re and E. In this case, error exponentially increases as we decrease Re below 100.

4.4.3 Noise on Error

In this Section, we will look at the effect that noise has on the performance of the preSTLS algorithm. It was suggested that the turbulent state is most probably on a chaotic saddle and not an attractor [3], i.e. the turbulent state will in time settle back onto the laminar state. This means that there will exist a turbulence lifetime that is dependent on the parametric changes. As discussed in earlier sections, varying parameters such as Reynolds number, Domain size and initial perturbation energy all have an effect on the turbulence lifetime of the flow. The performance of the preSTLS algorithm relies on the abundance of turbulent state data, with the extracted coefficients being more accurate the more turbulent data we have. Furthermore, the additive Gaussian noise greatly impacts the performance of the preSTLS algorithm. Therefore, it can be suggested that the performance of the PreSTLS algorithm is dependent on the abundance of turbulent data (turbulent lifetimes) and the snr value of the added noise. We will compare the effect of noise and survival lifetimes on the error for the 10x10x10 data extraction example, this gives Figure 22.

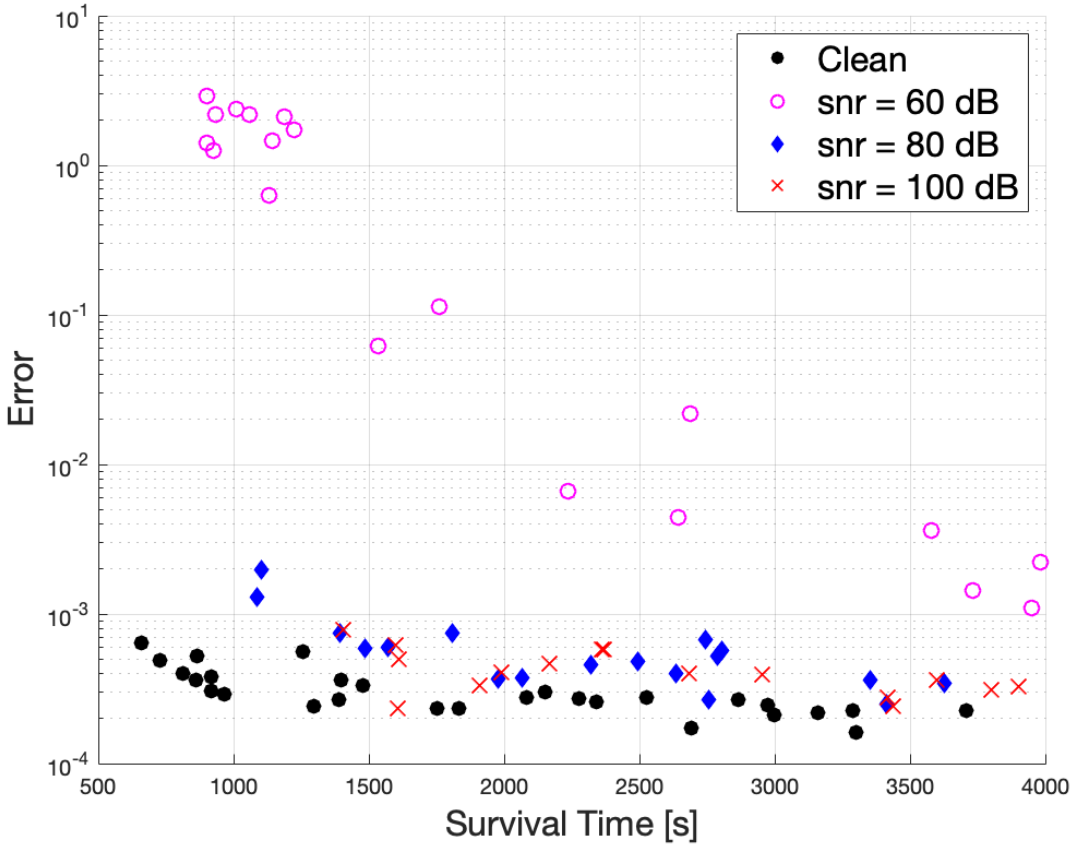


Figure 22: Comparison of error against turbulence lifetime for different snr values. Using 10x10x10 points extraction.

From Figure 22, the clean data points are the same as those in Figure 20a. Here the error is defined as the L^1 norm of the difference between the exact and extracted ODE coeffcients. For relatively low noise levels, or high snr values of 100 dB and 80 dB, the error distribution is similar to the case with the clean data. However, for relatively high noise, or a low snr value of 60 dB, we observe a steeper error increase as survival time decreases, and for survival times around 1000s we see a cluster in data points. This cluster in data point represents the failure in the preSTLS algorithm. In Figure 10, we also looked at the effect of a 60 dB snr value, and in that example, the survival time was around 4000s, hence the successful capture of the ODE coefficients.

The failure of the preSTLS algorithm can be attributed to the least squares extraction of state data from noisy velocity data. An example of failed state extraction can be seen in Figure 23.

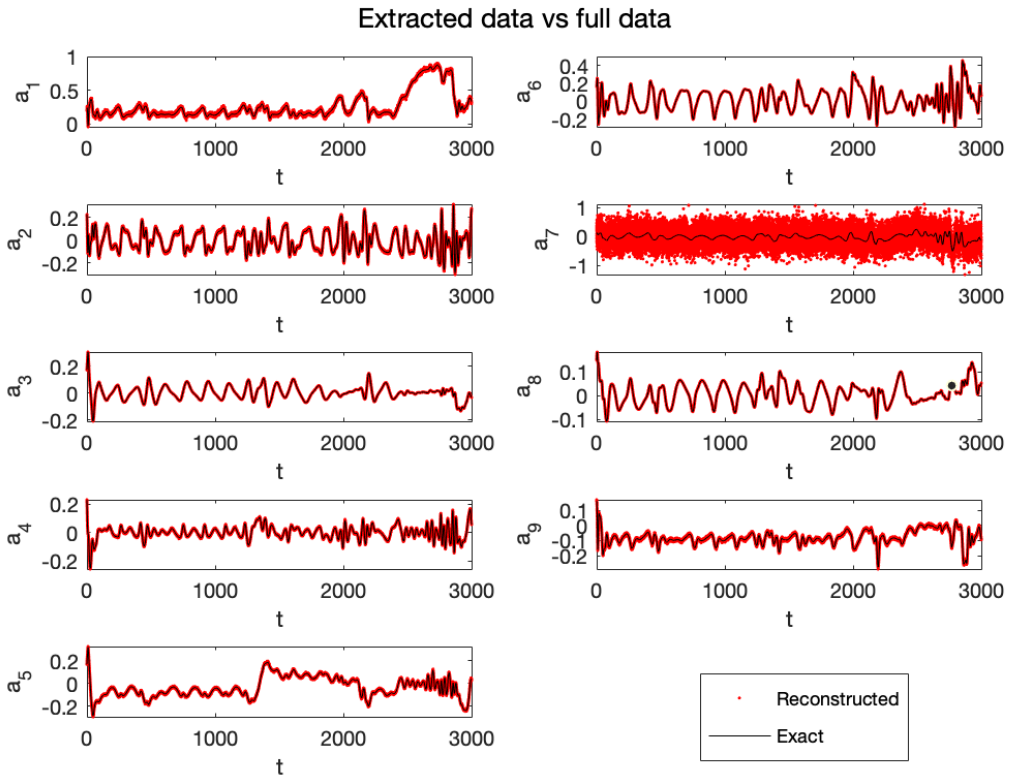


Figure 23: States plot with its extracted values from 10x10x10 velocity data points, showing failure at an snr of 60 dB and survival time of 1100s.

5 Conclusion

The analysis of the performance of the STLS algorithm was conducted using the velocity data extracted from the low-dimension turbulent shear flow model. The states data $\mathbf{a}(t)$ were extracted from the velocity data using least squares, which gives the preSTLS algorithm.

At relatively low noise, the preSTLS algorithm works well, but at lower snr values the least squares step in extracting states data from velocity data fails. The analysis in Section 3.3 was conducted using states estimation from velocity data extracted from 5 sensor points. This shows that it is possible to accurately infer states data from velocity data using only 5 sensor points for low noise.

An exponentially decaying relationship was identified between the lifetime of the turbulent regime and the error in the identified ODE coefficients using STLS. The lifetime of the turbulent regime was found to be dependent on the initial perturbation energy and Reynolds number in a fractal manner. From [3], it was suggested that the turbulent state is not an attractor but an open hyperbolic structure, with the chaotic saddle present at higher Reynolds number. With the lower Reynolds number, all shear flows for different initial conditions decay to the laminar profile. It is difficult to show the existence of a chaotic saddle but perhaps more complex dynamics could be identified when implementing Algorithm 2 on wind tunnel or DNS data for a bigger candidate function library and using different spatial modes.

In reality, the small perturbations from noise would change the trajectory of the states as it happens, but in this paper it was assumed that this change in trajectory is accounted for in the low dimensional shear flow model, and additional white Gaussian noise is added to test the robustness of the algorithm. Perturbation analysis was conducted by [3], and in a future paper it would be interesting to see how the preSTLS algorithm performs with perturbed states data.

Furthermore, because velocity data came from the model itself, thus the identified coefficients would be biased towards the flaws of the model. So if actual wind tunnel or DNS data were to be used, the algorithm could return the underlying ODE that would capture different modes and more complex coupling. On another note, there is a sparse regression method similar to the STLS, called Sequential Thresholded Ridge Regression (STRidge) [25], whereby ridge regression is used instead of least squares. Perhaps in a future paper, it would be good to explore this method and its performance in identifying underlying ODEs of turbulent shear flows.

The development of low-dimensional models to describe turbulent flows is important in fluid-dynamics, and it has enabled complex analysis in many industries ranging from aerospace to medicine. With increasing computational resources and an abundance of data with no associated underlying model, the development of data-driven discovery of system dynamics is becoming more important and necessary.

References

[1] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proceedings of the National Academy of Sciences, 113(15):3932–3937, 2016.

[2] Jeff Moehlis, Holger Faisst, and Bruno Eckhardt. A low-dimensional model for turbulent shear flows. New Journal of Physics, 6, 2004.

[3] Jeff Moehlis, Holger Faisst, and Bruno Eckhardt. Periodic orbits and chaotic sets in a low-dimensional model for shear flows. SIAM Journal on Applied Dynamical Systems, 4(2):352–376, 2005.

[4] Victor Emmanuell Badea, Alin Zamfiroiu, and Radu Boncea. Big data in the aerospace industry. Informatica Economica, 22(1/2018):17–24, 2018.

[5] J. Bongard and H. Lipson. Automated reverse engineering of nonlinear dynamical systems. Proceedings of the National Academy of Sciences, 104(24):9943–9948, 2007.

[6] J. S. Shang. Computational fluid dynamics application to aerospace science. The Aeronautical Journal, 113(1148):619–632, 2009.

[7] R.W.C.P. Verstappen and A.E.P. Veldman. Direct numerical simulation of turbulence at lower costs. Journal of Engineering Mathematics, 32(2/3):143–159, 1997.

[8] Peter J Schmid. Stability and transition in shear flows. Springer, 2012.

[9] Fabian Waleffe. On a self-sustaining process in shear flows. Physics of Fluids, 9(4):883–900, 1997.

[10] Fabian Waleffe. Homotopy of exact coherent structures in plane shear flows. Physics of Fluids, 15(6):1517, 2003.

[11] J. Moehlis, T. R. Smith, P. Holmes, and H. Faisst. Models for turbulent plane couette flow using the proper orthogonal decomposition. Physics of Fluids, 14(7):2493, 2002.

[12] Ajay Kumar Boyat and Brijendra Kumar Joshi. A review paper : Noise models in digital image processing. Signal & Image Processing : An International Journal, 6(2):63–75, 2015.

[13] Jun Xu, Lei Zhang, and David Zhang. External prior guided internal prior learning for real-world noisy image denoising. IEEE Transactions on Image Processing, 27(6):2996–3010, 2018.

[14] Ajay Boyat and Brijendra Kumar Joshi. Image denoising using wavelet transform and median filtering. 2013 Nirma University International Conference on Engineering (NUiCONE), 2013.

[15] Laurent El Ghaoui, Vu Pham, Guan-Cheng Li, Viet-An Duong, Ashok Srivastava, and Kanishka Bhaduri. Understanding large text corpora via sparse machine learning. Statistical Analysis and Data Mining, 6(3):221–242, 2013.

[16] Bhawna Dhruv, Neetu Mittal, and Megha Modi. Analysis of different filters for noise reduction in images. 2017 Recent Developments in Control, Automation & Power Engineering (RDCAPE), 2017.

[17] Bernard W. Silverman and Iain M. Johnstone. Needles and straw in haystacks: Empirical bayes estimates of possibly sparse sequences. The Annals of Statistics, 32(4):1594–1649, 2004.

[18] 2019.

[19] N. R Draper and Harry Smith. Applied regression analysis. Wiley, 1981.

[20] Simon Foucart and Holger Rauhut. A Mathematical Introduction to Compressive Sensing. Birkhašuser, 2013.

[21] Geoff Boeing. Visual analysis of nonlinear dynamical systems: Chaos, fractals, self-similarity and the limits of prediction. Systems, 4(4):37, 2016.

[22] R. E. Cline and R. J. Plemmons. L2-solutions to underdetermined linear systems. SIAM Review, 18(1):92–106, 1976.

[23] James M. Hamilton, John Kim, and Fabian Waleffe. Regeneration mechanisms of near-wall turbulence structures. Journal of Fluid Mechanics, 287(1):317, 1995.

[24] R. M. Clever and F. H. Busse. Tertiary and quaternary solutions for plane couette flow. Journal of Fluid Mechanics, 344:137–153, 1997.

[25] Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Data-driven discovery of partial differential equations. Science, 3(4):e1602614, 2017.

Appendix A: Indices for Trial Functions

Index	Trial Function	Index	Trial Function
1	1	29	yz^2
2	x	30	y^2z
3	y	31	yz^3
4	z	32	y^3z
5	x^2y^2	33	yz^4
6	x^2y^3	34	y^4z
7	x^3y^2	35	x^2
8	x^2z^2	36	x^3
9	x^2z^3	37	x^4
10	x^3z^2	38	x^5
11	y^2z^2	39	y^2
12	y^2z^3	40	y^3
13	y^3z^2	41	y^4
14	xy	42	y^5
15	xz	43	z^2
16	yz	44	z^3
17	xy^2	45	z^4
18	x^2y	46	z^5
19	xy^3	47	xyz^2
20	x^3y	48	xy^2z
21	xy^4	49	x^2yz
22	x^4y	50	xyz^3
23	xz^2	51	xy^3z
24	x^2z	52	x^3yz
25	xz^3	53	xy^2z^2
26	x^3z	54	x^2yz^2
27	xz^4	55	x^2y^2z
28	x^4z	56	xyz

Table 4: Trial functions for the Lorenz system and their respective indices

Index	Trial Function	Index	Trial Function
1	1	24	a_2a_8
2	a_1	25	a_2a_9
3	a_2	26	a_3a_4
4	a_3	27	a_3a_5
5	a_4	28	a_3a_6
6	a_5	29	a_3a_7
7	a_6	30	a_3a_8
8	a_7	31	a_3a_9
9	a_8	32	a_4a_5
10	a_9	33	a_4a_6
11	a_1a_2	34	a_4a_7
12	a_1a_3	35	a_4a_8
13	a_1a_4	36	a_4a_9
14	a_1a_5	37	a_5a_6
15	a_1a_6	38	a_5a_7
16	a_1a_7	39	a_5a_8
17	a_1a_8	40	a_5a_9
18	a_1a_9	41	a_6a_7
19	a_2a_3	42	a_6a_8
20	a_2a_4	43	a_6a_9
21	a_2a_5	44	a_7a_8
22	a_2a_6	45	a_7a_9
23	a_2a_7	46	a_8a_9

Table 5: Trial functions for the low dimensional turbulent shear flow model and their respective indices