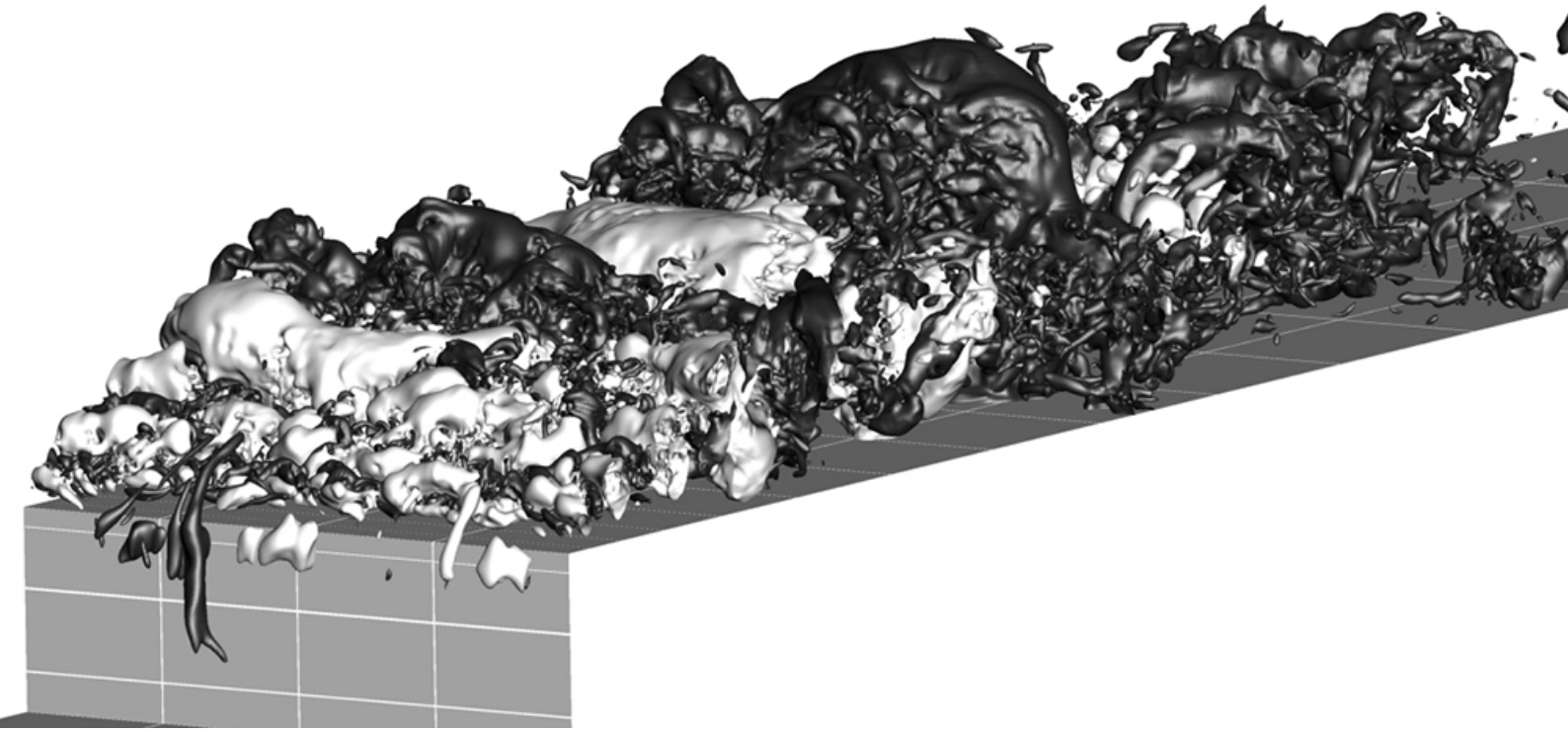
Subgrid-Scale model using Artificial Neural Networks for Wall-bounded Turbulent flows

A reseach on the fidelity of the model

Naina Kurian

January 22, 2018



TU Delft

Subgrid-Scale model using Artificial Neural Networks for Wall-bounded Turbulent flows

A reseach on the fidelity of the model

by

Naina Kurian

in partial fulfillment of the requirements for the degree of

Master of Science
in Aerospace Engineering

at the Delft University of Technology
to be defended on Wednesday January 29, 2018 at 13:30.

Student number:	4920547	
Supervisor :	Dr. S. J. Hulshoff	TU Delft
Thesis Committee :	Dr. ir Bas Van Oudheusen	TU Delft
	Dr. R. P. Dwight	TU Delft
	Dr. S. J. Hulshoff	TU Delft
	Dr. ir E. Van Kampen	TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Preface

This thesis was done as a part of the Master's curriculum in TU Delft. My life as a Master's student in TU Delft has been an enriching experience. They say you only grow as a human being if you're outside your comfort zone While my knowledge has only abundantly grown in these two years, it has also made me a responsible, independent human being. At this point of time, when I'm slowly reaching the end of my master's life, I would like to acknowledge those people who have made this journey easier for me.

I would like to thank my loving parents for all their support throughout my life. Without their undeterring faith in my abilities, I would not have been able to make it this far. I would also like to extend this gratitude to the rest of my family who have constantly propelled me to reach greater heights. A special thank you to my dear cousin Kavya, my schengen sister. If you weren't taking this step of coming to study in europe alongside me and supporting me throughout it, it would have made this whole experience much harder for me.

I would also like to thank my supervisor, Dr Steven Hulshoff. He has helped me in my first year, when I was troubled with my study. He took up the role of the track coordinator and directed me in the right path. Later on, as my supervisor, he has been helpful in every way and I have learnt so much in this one year from him. Thank you, Sir.

A special thank you to Satya, who's been my constant support, through my unpredictable mood swings, my worries with masters, my sadness of being unable to go home. With so much ease, you turned my frown around. Thank you. My friends here have also helped me go through these difficult times with ease. Yeshwanth, Greeshma, Deepak and Manju are a few to name. We've all been a support to one another, and its hard to imagine these years without all of you. Thank you all of you, for being my family away from home.

Naina Kurian

Delft, December 2017

Abstract

Werner Karl Heisenberg, a renowned German physicist is reported to have said :

> 'When I meet God, I am going to ask him two questions: Why relativity? And why turbulence? I really believe he will have an answer for the first'

The chaotic nature of turbulent flows combined with the diverse scales present within turbulent flows makes these flows very challenging to accurately simulate. Since turbulence is a phenomenon that cannot be avoided in real flows, a good understanding and accurate representation of turbulent flows would help to advance in the technical front in a sustainable manner. Over the years, LES and DNS have come up as different methods by which this problem of turbulence could be solved, to an extent. DNS resolves all the relevant scales thereby providing the most accurate option. However, it can only be used for flows with low reynolds number. LES is a less costly option but it requires a wall model in order to represent the wall bounded flow with practical computing costs. An attempt was made by Durieux [10] as well as Beekman [3] to create such a model for the 1D burgers equations, with the help of Artificial Neural Networks(ANN). In both cases, the ANN was used to find unresolved scale model coefficients that are the closest fit so that the results of the LES are close to the ones obtained by a DNS. In order to do this efficiently, a training error measure(TEM) was developed which was used by the ANN to obtain the correct model coefficients. This worked well, but only one form of unresolved scale model (algebraic Variational multiscale model) was used and the cases considered were easily handled by that particular model.

The aim of this thesis is therefore to examine the effectiveness of the procedures used so far for alternative subgrid scale forms of the unresolved scale model and more challenging test problems. In particular, the ANN calibration of the Orthogonal subgrid scale model and the Dynamic Orthogonal subgrid scale model are considered using test cases of constant forcing, periodic forcing and a Stochastic forcing. The stochastic forcing case would provide an initial idea of how the results for Navier stokes would look like. Response surfaces for both the training error measure(TEM) as well as the simulation error measure (SEM) were also plotted to further investigate the performance of the ANN.

It is shown from this thesis that the ANN model is able to work with different forms of the unresolved scale model and different forcing functions. Additionally, the potential gains resulting from spatially varying coefficients are demonstrated. It was also found that the training error measure(TEM) used to train the ANN is able to work efficiently only if there exists model coefficients that would fit the reference data correctly. In a scenario where these coefficients do not exist, the training error measure(TEM) can behave very differently from the simulation error measure(SEM) which results in the ANN producing results that are very different from what is required. A major conclusion is therefore that while the TEM is a good measure to qualitatively analyse the situation, in challenging cases it would be worthwhile to consider a slightly different approach.

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Networks
ASGS	Algebraic Subgrid Scale
BC	Boundary Conditions
DDES	Detached Delayed Eddy Simulation
DES	Delayed Eddy Simulation
DNS	Direct Numerical Simulation
DSS	Dynamic Subgrid Scale
IC	Initial Conditions
LES	Large Eddy Simulation
OSS	Orthogonal Subgrid Scale
RANS	Reynolds Averaged Navier Stokes
SEM	Simulation Error Measure
SGS	Subgrid Scale
TEM	Training Error Measure
TLM	Two Layer Method
VLSM	Very Large Scale Motions
WMLES	Wall Modeled Large Eddy Simulation

Nomenclature

$\bar{u}$	Velocity of resolved/larger scales
γ_1, γ_2	Model Coefficients
$\mathscr{P}$	L^2 projection into the resolved scale space
$\mathscr{R}$	Residual
ν	Viscosity of flow
f	Forcing term in the burgers equation
k	Wave Number

u'	Velocity of unresolved/smaller scales
w	Test function for the multiscale framework

1

Introduction

The prediction of turbulent flows is important for many fields including airplane performance, weather prediction, space programs, wind loading on infrastructure, oil rigs, engines and so on. Therefore the ability to accurately simulate turbulent flows is very useful to science and engineering. The main difficulty that rises is the wide range of scales that are present within turbulent flow. Direct numerical simulation(DNS) aims to fully resolve even the smallest scales of turbulence. According to theory, if the largest scales are denoted by L and the smallest scales(kolmogorov) are denoted by η, then the ratio of the two is given by equation 1.1.

$$\frac{\eta}{L} \sim Re^{-3/4} \tag{1.1}$$

This makes DNS computationally infeasible for high reynolds numbers. This problem only worsens when the flow is in the vicinity of a solid boundary. This is because the length scales of turbulent structures decrease rapidly as the flow approaches the solid boundary.

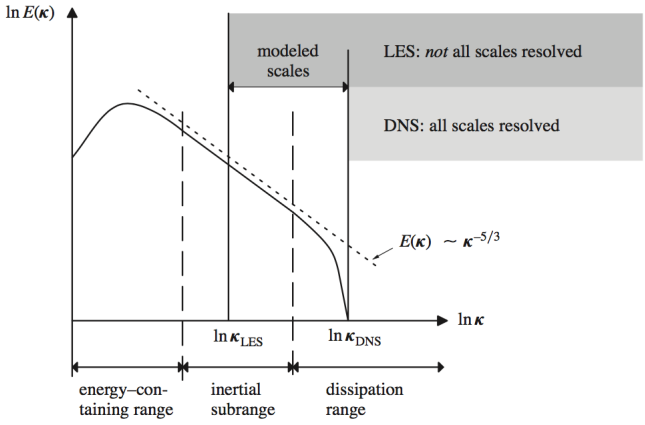


Figure 1.1: DNS and LES in the Energy spectrum Gravemeier et al. [12]

An alternative to DNS is then Large Eddy Simulation (LES). Here only the larger scales are resolved and the smaller scales are modelled. The advantage of LES is the use of a coarser grid which is computationally efficient. LES relies on the Kolmogorov hypothesis which assumes that the smaller scales have a more universal character. In LES, one models the small scale influence with simpler numerical methods. The major difference between DNS and LES can be seen in the energy spectrum seen in figure 1.1. Classical LES models however, tend to be inaccurate for wall bounded turbulence. In order to increase the accuracy, finer grids need to be employed which increases the costs of an LES calculation. Wall bounded turbulence, therefore needs a subgrid-scale (SGS) model that would

be both accurate and cost effective. In a world where machine learning is winning over many fields of sciences, its application to this problem could be the answer. This research involves the detailed study of one such model to test its fidelity to different conditions.

1.1. The need for Wall models

As Re increases, the smallest scales of turbulence decreases significantly, thus causing severe challenges in the study of these flows (Metzger and Klewicki [27]) .These small scales of turbulence poses challenges for the spatial resolution of the sensor in experimentation thus necessitating the need for alternate techniques. (Metzger and Klewicki [27]). Numerically these flows can be solved with good accuracy but with high amount of computational cost. This is owing to the fine mesh size in the near wall region. Piomelli [32] shows that at high reynolds numbers, more than 50 % of the computational cost is for the inner layer alone. It should also be noted that most technologically significant flows, especially the ones in the atmosphere, are at a high reynolds number (Re>100000). This warrants the need for new models that can compute challenging turbulent flows at high reynolds number more efficiently. Research on this area has been increasingly active in recent years (Marusic et al. [24], Meneveau and Katz [26]).

1.1.1. Existing Wall Model Techniques

A large number of wall modelling techniques have been suggested by multiple researchers to overcome the challenges in wall bounded turbulent flows (Marusic et al. [24]). Since most of the difficulty arises from the inner layer because of the requirement of dense computations, most work has been done in computing flow in this layer effectively. The time taken for computations in which the inner layer is resolved, scales with $Re^{2.4}$ but a notable fact is that the time taken for computations in which only the outer layer is resolved, scales with only $Re^{0.8}$. Thus, it is desired to fully resolve this outer layer ehile modelling the inner layer.. This kind of approach is known as Wall-Modeled LES (WMLES). Piomelli [32] states that there are mainly three kinds of WMLES calculations.

- **Wall stress models** : Empirical formulae are used to connect the inner layer characterestics to the dependent variables on the first grid of the outer layer where a LES calculation is performed. The main advantage of these models is the cost effectiveness However,they are not applicable to all engineering problems, since they all assume a logarithmic layer where the stress is assumed to be constant Piomelli [32]. It must also be mentioned that Hutchins and Marusic [19] had found evidence of 'superstructures' in the logarithmic region of a turbulent boundary layer. These structures are now termed as Very Large scale motions (VLSM) and researchers aim to estabilish the relationship between these VLSM and the inner layer with the help of mathematical models. Hutchins and Marusic [19] further explained that the VLSM structures are meant to be resolved and they must inner layer must react to the VLSM. In other words, the inner layer must be shown as if it was "blown on" by the VLSM. This discovery has helped improve wall models because the VLSM's have an effect on the fluctuations near the wall, which means that computing the VLSM (which is a on a much coarser grid) is enough if the relation between the VLSM and the inner layer fluctuations can be modelled. This type of a wall model has been achieved by [24]. Another type of wall stress model incorporates is the use of boundary conditions to redefine the near-wall requirements so as to ensure appropriate behaviour of the flow (Piomelli [32], Hulshoff et al. [17]).

- **Two Layer Models** : The most known method that falls in the broad section of zonal approaches is the Two Layer Method (TLM). This method requires a simplified equation which is solved within the inner layer using a fine one dimensional embedded grid. An LES calculation is performed on the outer layer. The inner layer calculations and the LES of the outer layer are

weakly coupled. The wall stress is found from the inner layer calculation which inturn uses the boundary condition obtained from the LES calculations. Good results were obtained by Balaras et al. [2] with an additional computing time of only 10-15 percent of the overall computing time. However, these methods are slightly ineffective when the origin of the eddies are within the inner layer.

- **Hybrid Methods** : In this method, the inner layer is computed by using RANS equations and the outer layer is computed by an LES simulation. These methods are rather expensive but versatile in nature. The main problem that arises with these hybrid computations is the consistency inaccuracies especially in the interface of the two layers owing to the different length and time scales. This inaccuracy can be clearly seen in the study conducted by Nikitin et al. [30] where the skin friction was underestimated by 15 percent. This error is estimated to be present due to the presence of eddies in the interface region (Baggett [1]). Research on methods to eliminate this error and blending the two layers in a more consistent manner, is ongoing. A method was proposed by Shur et al. [39] where a delayed detached eddy simulation (DDES) is combined with an improved hybrid RANS-LES model. The main disadvantage with using RANS based model is that it produces mean values and thereby no instantaneous information about the flowfield is available, making the results incomplete. This results in a log layer mismatch, which could be removed by altering the DES equations with empirical improvements.

Most of the problems with these models described above deals are related to their coupling with the LES calculation. As mentioned within the wall stress models subsection, the relationship between the VLSM and the inner layer was used by Marusic et al. [24] to build a successful model. In a similar manner, a model could be built with the help of artificial neural networks which could mimic the inner region effects and provide its influence on the outer layer as an output.

1.1.2. Artificial Neural Network based subgrid scale model

Artificial neural networks are computational tools which are being used currently in pattern recognition, language processing, speech recognition, medical diagnosis and various other fields of science and technology. Broadly it can be said that the neural network is used to solve two main types of problems, regression and classification. In the current scenario, where the neural network is in itself a subgrid scale model for turbulence modelling, it works for a regression problem. In some ways, this problem can be considered to be a complex curve fitting problem which is solved by the neural network.

In the field of fluid dynamics and its research, Artificial neural networks (ANN) is a relatively new member. However its use has been extensive and successful. In turbulence research, ANN's have been used to predict the coefficients of Double Proper Orthogonal Decomposition in a turbulent channel (Cohen et al. [9]). Time Delay Neural Networks (TDNN) have been used to predict the values from hot wire measurments by Duro and Santos [11] and Peña et al. [31]. In all this research it was shown that ANNs are capable of predicting even highly complex flows like ones which can be seen in turbulence. However for wall bounded turbulent flows, the complexity increases and the use of ANN's in such a scenario is not well explored. One of the research involves the work done by Sarghini et al. [36] and Vollant and Corre [42] where ANN based SGS models were implemented using the filtering operation for LES calculations. These works have shown promise with respect to the introduction of ANN's within Subgrid scale models.

An interesting alteration to this, is the use of the Variational Multiscale Method in place of the traditional filtering operations within LES calculations. An initial study of this topic was done by Durieux [10] in which the model equation was taken as the Burgers equation which is a simplified form of the Incompressible Navier Stokes(INS) equation. This study showed an increase in performance of

the model. The simulation error measure (SEM) is the difference between the reference values(in this case, DNS results) and the values from the LES calculation. In order to decrease the computational cost,Durieux [10] devised a Training Error Measure(TEM) which is used to train the neural network.Following Durieux [10], Beekman [3] developed a model using the burgers equation which was flexible and could be extended to more complex problems such as a two or three dimensional problem based on the Navier stokes.His results have also shown good improvement in performance and is the starting point of the current study. The TEM has worked well for the cases considered by Durieux [10] and Beekman [3]. This study aims at developing insights for the 1-D problem better by using different forms of the unresolved scale models(namely the orthogonal subgrid scale model and the dynamic orthogonal subgrid scale model) as well as using different forcing terms to understand the effect the ANN has if the data is more nonlinear in nature. In other terms the goal of this thesis is to test the range of the procedures developed for constructing ANN-based SGS models.

1.2. Research questions

The near-successful implementation of the ANN based SGS model by Beekman [3] paves the path for a new LES model which could be more efficient for the case of wall bounded flows. However this model was built for the Burgers equation and was tested for only a few cases. While the results were promising, it is still not understood as to how the ANN functionality varies with different flow conditions. Therefore it is necessary to understand the fidelity of the model and make the necessary changed to its implementation before it can be efficiently applied to a incompressible Navier stokes channel flow.

Research objective

"**Perform network training with different forms of the unresolved scale model and different forcing terms so as to test the fidelity of the ANN-based SGS model under different conditions**"

Several questions were formulated to help achieve this research objective.

- *Does the ANN based SGS model work well even for cases with high nonlinearity?*

  Beekman [3] explored the effect of this model with both constant forcing as well as periodic forcing. Another way of assessing the performance of the ANN is to attempt to use a stochastic forcing term in the Burgers equation which would mimic the pressure term of the Navier Stokes and induce stronger non linearity. The behaviour of the ANN under such conditions is crucial, because this would give a first look to how the ANN may perform with Navier Stokes.

- *Does changing the model affect the accuracy and training method required?*

  While the algebraic form of the unresolved scale model is the most used one, Guasch and Codina [13] used an orthogonal form which seems to have obtained better results in the multiscale framework. The same research also used a dynamic form, where the effects of the transient term was also included in the subgrid scale calculations so as to get more reliable results. This residual is then projected orthogonally for the Dynamic Orthogonal Subgrid scale model. Ravichandran [34] also proved that the orthogonal form showed improvements. Therefore it would be worthwhile to see the performance of the ANN-based SGS model with different forms of the unresolved scale model. This result could help make a more versatile model which could produce the best results depending on the nature of the problem.

- *Does the ANN based SGS model work better than the LES with standard coefficients under different conditions? If not what could the reasons be?*

  The results of Beekman [3] proved that under most cases, the ANN based SGS model outperforms the LES with standard coefficients. However inducing higher non linearity and the effect of the different forms of the unresolved scale model, might give an alternate perspective. Exploring the reasons for this could help build a better network which could be capable of outperforming the LES under all conditions.

- *Is the Training error norm effective for all the cases considered? If not,what changes need to be made to make it more universally correct?*

  The training norm considered in the current model is the squared sum of the weak residual per test function, averaged over the different time steps. For the highly non linear cases it could be possible that the training norm is not effective. It could also be possible that the neural network in itself requires a better architecture or more input nodes. This information could eventually help build a neural network which would work efficiently for the Incompressible Navier stokes case.

- *Would the effect of spatially varying coefficients be large or small?*

  While the model coefficients changing in space would be a more realistic model, the effect of this was not shown to be significant through results by Beekman [3]. It would be worthwhile to investigate the effect of this modelling procedure as it would expose the true capabilities of the neural network.

1.3. Outline of this thesis

This thesis contains the relevant theory as well as the results found through the investigation done by the author. Chapter 2 explains the variational multiscale method which is the backbone of the fluid simulations. This method is used as an alternative to the classical filtering approach. Within this chapter, the variational multiscale framework is detailed along with its application to the burgers equation. The different 2 scale SGS models used in this thesis and their relevance is also sufficiently explained accompanied with equations in the variational form within the same chapter. Chapter 3 dives into Artificial Neural networks and their use as a subgrid scale model. The performce terms used for training are discussed in detail within this chapter. The next chapter discusses the computational setup. It explains the layout of the model with a flowchart. Chapter 5 analyses the results obtained from this study for various test cases. The conclusions are drawn in chapter 6, which is followed by recommendations in chapter 7.

2

Multiscale Methods

Classically, LES calculations involved a filtering technique, which was required in order to separate the scales which needed to be resolved from the scales which are modelled. In classical LES, within the filtered equations, the subgid scale stress are modelled after which the scales are separated '*a posteriori*'. The variational multiscale method(VMM) is an alternative which was originally a theoretical framework developed by Hughes et al. [15], and later used for general practical problems by Hughes et al. [16]. It was pointed out through his study that the method could be used to project the dynamics of the unresolved scales into the resolved scales. As a contrast to classical LES which uses spatial filters for scale separation, the scales are defined by projection into appropriate spaces. This method has several advantages over the classical LES method. The variational multiscale method yields the correct asymptotic behavious near the walls.VMM uses an '*a priori*' scale separation which helps to resctrict modelling to flow related to the high wavenumbers only. This also ensures that the numerical errors and modelling errors are united. It is able to predict backscatter which was one of the shortcomings of the classical smagorinksy model.The ANN-based SGS model works on the variational multiscale framework. In the next section, the application of the VMM to a general problem is explained.

2.1. VMM Framework

The variational multiscale method employs a framework for subgrid scale modeling which can be explained with the help of a general problem,as discussed below. :

$$\mathscr{L}(u) = f \qquad in \ \ \Omega \tag{2.1a}$$

$$u = g \qquad on \ \ \Gamma \tag{2.1b}$$

where $\mathscr{L}$ is a differential operator, u is an independent variable and f is the forcing term. Here Γ denotes the boundary of the domain which is Ω. Let $\mathscr{S}$ denote the solution space and $\mathscr{V}$ denote the weighting space such that

$$(u) = g \qquad \forall u \in \mathscr{S} \tag{2.2a}$$

$$w = 0 \qquad \forall u \in \mathscr{V} \tag{2.2b}$$

This problem,as expressed in equation 2.1, is then represented in its variational form, by multiplying it with a test function w.

$$\int_\Omega w\mathscr{L}(u)d\Omega = \int_\Omega wfd\Omega \tag{2.3}$$

For ease of expression, a notation as followed by Hughes et al. [16] will be used. This notation is explained by equation 2.3, where the integrals are shortened using brackets and subscripts. Hereafter this notation will be used.

$$(w,u)_\Omega = \int_\Omega wud\Omega \tag{2.4}$$

It is in common practice within the subject of turbulence to decompose the flow variables into the mean values and the fluctuations. In similar lines, the solution is partitioned as $u = \bar{u} + u'$, where $\bar{u}$ denotes the large scales and u' denotes the smaller scales. The solution space and weighting space is also decomposed as $\mathscr{S} = \bar{\mathscr{S}} + \mathscr{S}'$ and $\mathscr{V} = \bar{\mathscr{V}} + \mathscr{V}'$ respectively.

$$\bar{u} = \sum_i^N a_i\phi_i \tag{2.5}$$

An LES computation, as mentioned before aims at getting a good representation of the $\bar{u}$ which contains the effects of u'. In order to solve for $\bar{u}$, the Galerkin solution procedure can be used. By this method the solution can be approximated by the equation 2.3. In order to solve this equation, the values of a_i are found by writing equation 2.5 for N test functions. The weighting space is also partitioned as $w = \bar{w} + w'$ where $\bar{w}$ is finite dimensional and contains the N functions which are necessary. On the basis of the linear independence of $\bar{w}$ and w', equation 2.4 can then be split into two equations, one pertaining to the large scales and the other, to the small scales.

$$(\bar{w},\mathscr{L}(\bar{u}+u'))_\Omega = (\bar{w},f)_\Omega \tag{2.6a}$$

$$(w',\mathscr{L}(\bar{u}+u'))_\Omega = (w',f)_\Omega \tag{2.6b}$$

Still, equation 2.6a has a missing link.The solution of u' is necessary for the resolved scale solution.While an unresolved scale equation can be written as shown by equation 2.6b, it is possible to solve this equation only with infinite resources, for all functions contained within w' which is a infinite-dimensional set. (S.J.Hulshoff [40]). Thus, the influence of the unresolved scales within the resolved scales has to be approximated.This is done by incorporating different types of numerical/analytical methods. This can be seen for the particular case of burger's equations as well.

2.1.1. Burgers Equation

The burgers equation is a simplified form of the Navier stokes equations(NSE). The 1D Burgers equation is given by equation 2.7 and is considered to have most of the factors that are studied in a turbulent flow. Since it is simpler and can be solved with lesS computational effort and time, it serves as a good starting point to study any new method.

$$u_t + uu_x - \nu u_{xx} = f(x,t) \tag{2.7}$$

In the variational framework, the large scale and small scale equations for the burgers equation can be given by equation 2.8a and equation 2.8b.

$$(\bar{w},(\bar{u}_t+u'_t)) + (\bar{w},\mathscr{L}_b(\bar{u}+u')) = (\bar{w},f) \tag{2.8a}$$

$$(w',(\bar{u}_t+u'_t)) + (w',\mathscr{L}_b(\bar{u}+u')) = (w',f) \tag{2.8b}$$

where $\mathscr{L}_b(u) = uu_x - \nu u_{xx}$

Simplifying the unresolved scale equations we get,

$$(w', u'_t) + (w', uu'_x - \nu u'_{xx}) = (w', f - \bar{u}_t - u\bar{u}_x - \nu\bar{u}_{xx}) \tag{2.9}$$

In the equation 2.9, the solution is exact, however an approximation for u' is required to solve it. In order to find the effect of the unresolved scales on the resolved ones, the goal is to approximate equation 2.9 such that the end result is a problem which contains mostly the resolved scales. This approximation is usually carried out as a function of the resolved scales. The exact model differs for each formulation considered in this study and will be explained in sufficient detail within the individual formulations.

2.2. 2 scale SGS models

Ravichandran [34] showed that for increasing amplitude of random forcing, the orthogonal model showed significant improvement over the algebraic model. Ravichandran [34] also showed that the OSS model is superior in the case of 3D channel flow as well. These reasons form the motivation behind the selection of the OSS model as one of the formulations of the small scale influence which is tested with the ANN based SGS model. Mainly, there are two forms tested, the algebraic form and the orthogonal form. These will be referred as the algebraic subgrid scale model(ASGS) and the Orthogonal subgrid scale model(OSS) respectively. The main difference between the two SGS models lies in the space in which the subgrid scales are modelled to be present in. In order to understand the two models and their differences, it is important to define the space of subgrid scales.

Space of subgrid scales

Let us assume that the solution space is defined by S. Once a Galerkin approximation is performed and a finite element partition of the domain is done, the resolved scales would be present in a finite element space represented by, $\bar{S}$. Then the following equation can be written:

$$S = \bar{S} \oplus S'$$

The space S' is an space that is required to complete $\bar{S}$ in S. The space S' is called the space of subgrid scales. In other words, $\bar{U} \in \bar{S}$ and $U' \in S'$.

2.2.1. Algebraic subgrid scale model

This is the most used forms of the unresolved scale. For the Burgers case, equation 2.9 needs to be approximated. In order to do this a Fourier analysis was performed by Guasch and Codina [13] and the result is given by equation 2.10a.The residual of the resolved scales is given by equation 2.10b.

$$uu'_x - \nu u'_{xx} \approx \tau^{-1} u' \tag{2.10a}$$

$$\mathscr{R}(u;\bar{u}) = f - \bar{u}_t - u\bar{u}_x - \nu\bar{u}_{xx} \tag{2.10b}$$

$$\tau(u) \approx \tau(u_N); \qquad \mathscr{R}(u;\bar{u}) = \mathscr{R}(\bar{u};\bar{u}) = \mathscr{R}(\bar{u}) \tag{2.11}$$

With this approximation, we can rewrite the unresolved scale equation as given in equation 2.12a. Further simplying by assuming that the transient terms in the unresolved scales are neglected, equation 2.12b can be obtained.

$$u'_t + \tau^{-1} u' = \mathscr{R}(\bar{u}) \tag{2.12a}$$

$$u' = -\tau \mathscr{R}(\bar{u}) \tag{2.12b}$$

where $\mathscr{R}(\bar{u})$ is the residual of the large scales and τ is the average element green function.For the linear advection case, τ is numerically constructed by Shakib and Hughes [38] and is expressed by equation 2.13.This approximation while locally incorrect within the element, produces an exact solution for $\bar{u}$, under certain circumstances

$$\tau_{linAD} = \left[\left(\frac{2a}{\Delta x}\right)^2 + 9\left(\frac{4\nu}{\Delta x^2}\right)^2\right] \tag{2.13}$$

Since the linear advection equation gives exact results with this formulation,for the burgers equation, a good starting point would be, to build upon the linear form. Chen et al. [7] used an optimized greens function formulation, to determine an equation for τ. The formulation of τ by Shakib and Hughes [38] is used and free coefficients γ_1 and γ_2 are used to account for the non linear effects. Goal oriented, optimization techniques are used to determine γ_1 and γ_2. In this thesis, artificial neural networks are used to find these model coefficients. This forms of the unresolved scale is termed as the ASGS.

$$\tau_{chen} = \left[\gamma_1\left(\frac{u}{\Delta x}\right)^2 + \gamma_2\left(\frac{\nu}{\Delta x^2}\right)^2\right] \tag{2.14}$$

This formulation for τ_{chen} was used by Durieux [10] and Beekman [3], whose work serves as the base for the current research. Another method inspired from the works of Scovazzi [37] is the asymptotic expansion formulation, where the equation 2.12b itself includes the term which contains the residual squares(R^2). This method also has two free coefficients which are used to account for the non linearity and was implemented by Navarro [28].

2.2.2. Orthogonal subgrid scale model

The orthogonal subgrid scale model is an alternative to the algebraic subgrid scale. The difference is that the space used for the subgrid scales is different. By isomorphism of S' and then assuming that the subgrid scales are non conforming, the following results are found.

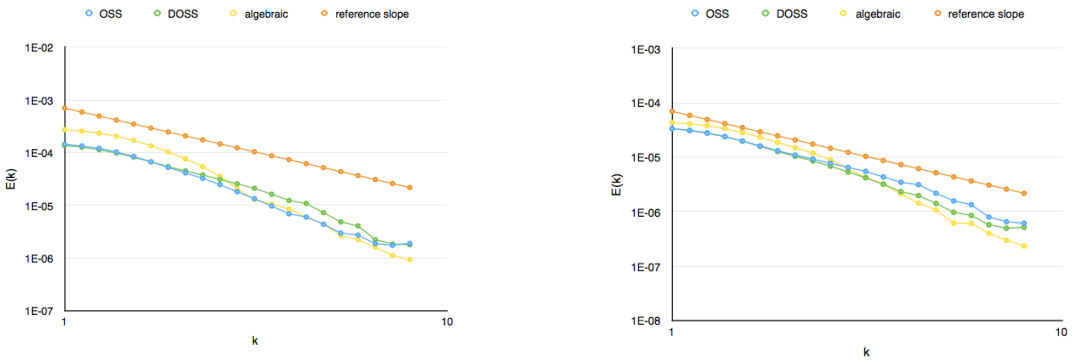
$$S' \simeq \bar{S}^{\perp} \cap S \tag{2.15}$$

$$S' \approx \bar{S}^{\perp} \tag{2.16}$$

The resulting equation for the subgrid scales is a function of the orthogonally projected resolved scale residual. This can be seen in equation 2.9 and 2.10. The orthogonally projected resolved scale residual is numerically equal to the difference between the local residual and the L^2 projection of the residual on to the appropriate finite element space.(Ravichandran [34]). For the quasi static case, the equation is simpler and the value for τ can be found by Fourier analysis. This model will be termed as OSS thoughout this thesis.

Why OSS?

Codina [8] explains that when the space defined by equation 2.9 is taken among the many possibilities, a stabilized numerical method can be created which uses the least squares form of components orthogonal to the finite element space considered. Codina [8] further shows that this new modelling strategy produces a method which is stable and has excellent accuracy. The computational cost of this method is higher than other methods for stationary problems however for transient and non linear problems, this method seems to form a promising alternative. Guasch and Codina [13] further performed a study to understand the behaviour of the stabilization terms used in this method. Results by Guasch and Codina [13] show that the OSS approach with transient subscales is able to predict backscatter, which was considered a flaw in the general Smagorinsky model. The OSS method



(a) Energy spectra for subgrid scale models with 2 × f

(b) Energy spectra for subgrid scale models with 4 × f

Figure 2.1: Comparison of energy spectra for different subgrid scale models with random forcing. (Ravichandran [34])

has shown promising results by Ravichandran [34] as shown by figure 2.1.The result that was provided by Ravichandran [34] showed that for increasing amplitude of random forcing, the orthogonal model showed significant improvement over the algebraic mode for larger range of wavenumbers.

Implementation of OSS in burgers equations

For the implementation of the Orthogonal subgrid scales, Guasch and Codina [13] explains that the subscales space is the space orthogonal to the resolved scales space. Therfore if $\mathscr{P}$ is considered to be the L^2 projection into the space of the resolved scales and $\mathscr{P}^{\perp} = I - \mathscr{P}$ where I is the identity, then the equation for the unresolved scales is given by equation 2.17a. If the transient terms of the unresolved equations are neglected in the OSGS model then the simplified equation is given by equation 2.17b.

$$u'_t + \tau^{-1} u' = \mathscr{P}^{\perp}\Big(\mathscr{R}(\bar{u})\Big) \tag{2.17a}$$

$$u' = -\tau\, \mathscr{P}^{\perp}\Big(\mathscr{R}(\bar{u})\Big) \tag{2.17b}$$

For the current study the value of τ is taken to be equal to the one seen in equation 2.14. The model coefficients are calculated by the ANN but this time for the case where the orthogonally projected residual is used i.e the OSS model.

2.2.3. Dynamic Orthogonal subgrid scale model

Guasch and Codina [13] also used a forms of the unresolved scale model where the transient term of the burgers equation is also taken into account for the subgrid scale equations. This forms a more realistic model. This model will be termes as DSS thoughout this thesis.

DSS in burgers equations

While the dynamic implementation can be used with both the algebraic and the orthogonal form of the unresolved scale model, for majority of this thesis, it is used with the orthogonal form of the unresolved scale model unless stated otherwise. For this case, the equation for the unresolced scales is given by equation 2.17a which is repeated below for better readability.

$$u'_t + \tau^{-1} u' = \mathscr{P}^{\perp}\Big(\mathscr{R}(\bar{u})\Big)$$

The results provided by Ravichandran [34] shown in figure 2.1 showed the DSS [1] to work the best in the case where the random forcing amplitude is doubled. This prompts the current author to

[1] Although Ravichandran [34] mentions DOSS, it is the same as DSS in the current scenario where the dynamic implementation is carried out with the orthogonal form of the unresolved scale model

incorporate the dynamic implementation within the ANN SGS model to see if the ANN is able to still provide reliable results.

Different implementations of DSS

For the sake of computational efficiency and to test the stability of the transient term, two different implementations of the DSS is carried out. One of them involves updating the trasient term every timestep whereas the other updates it locally in time. The latter implementation although more accurate, often leads to stability and convergence issues. In this research both the implementations are used. This would help conclude if the ANN can provide reliable converged results for a case when the term changes locally in time. For the purpose of separating the two implementations, the former one will be referred to as DSS1 and the latter as DSS2.

3

Artificial Neural Networks

Artificial Neural Networks are becoming an increasingly used machine learning technique in multiple fields of sciences and technology. ANN's have been used for mainly classification and regression problems. For a basic introduction to Artificial Neural Networks, the reader is encouraged to read the appendix-B. In this chapter, the neural network as used in the ANN SGS model is discussed. The performance term used by the ANN for its iterations is also explained. The different capabilities of the current network is also discussed.

3.1. The Neural Network in the ANN-SGS model

For the neural network considered, a Multi-Layer Perceptron(MLP) architecture is used. Since the current study works on testing the fidelity of the working model as devised by Beekman [3], the model was used with a similar architecture as Beekman [3] did. Moreover, the MLP structure was shown to provide improved results over the Fully Connected Cascade(FCC) by Durieux [10]. For all the input and output neurons a linear activation function is used and for the hidden neurons a hyperbolic tangent activation function is used. The x location is given as an input to the neural network. It was already mentioned in the previous chapter that the ANN attempts to find the model coefficients that are present within the equation of τ for each of the 2 scale SGS models. Therefore the output of this neural network is the two model coefficients γ_1 and γ_2. There were two cases considered with respect to the neural network layout.

3.1.1. Spatially constant model coefficients

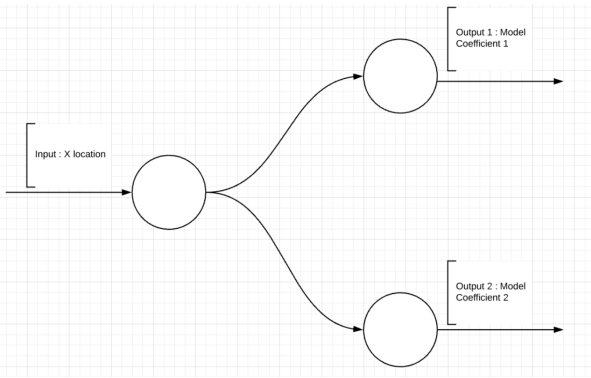


Figure 3.1: ANN-SGS model neural network for the spatially constant requirement

To ensure that the neural network works in the right manner, the neural network is made for a case where the model coefficients remain the same in space. It was shown by Beekman [3] that this case

also produces good results. Therefore this case is also used initially in this thesis to perform the fidelity investigation. For this case, the neural network is rather simple. A schematic of a neural network for this condition is given in figure 3.1. It can be seen that there is one input node and two output nodes which is a general requirement. However it is also seen that there are no hidden neurons. Moreover, for this case the weights of all the connections are kept as zero. Therefore only the effect of the bias is seen in the results.

Flow within the network

Initially, the x location is fed into the network. The weight of the connection to the input neuron is zero. The output from the input neuron is then given by Z_{input} given in equation 3.1 where f is the activation function and b is the bias of the neuron.

$$Z_{input} = f((w_1 \times x) + b_{input}) = f(b_{input}) \tag{3.1}$$

This output from the input neuron is taken as the input into the output neurons and the following equations can be written. Here w_{11-21} is the weight of the connection between the first layer,first neuron to the second layer, first neuron.

$$Z_{output1} = f((w_{11-21} \times Z_{input}) + b_{output1}) = f(b_{output1}) \tag{3.2a}$$

$$Z_{output2} = f((w_{11-22} \times Z_{input}) + b_{output2}) = f(b_{output1}) \tag{3.2b}$$

The $Z_{output1}$ and $Z_{output2}$ are the required γ_1 and γ_2 respectively. This case is the simplest in terms of the flow within the network, but it is important to know that this case works well with respect to the integration with the fluid model and the effect of the performance term used by the neural network. The performance term will be explained in the next section. But before moving into the integration and the performance term, a similar discussion is also shown for the case where the model coefficients are varied in space and the weights of the connections are no longer maintained at zero.

3.1.2. Spatially varying model coefficients

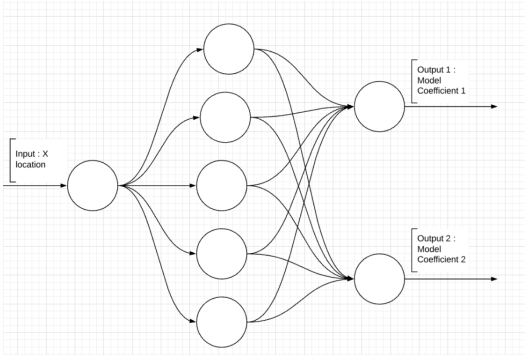


Figure 3.2: ANN-SGS model neural network for the spatially varying requirement

In real flows, the flow is inhomogenous and therefore the flow physics is different with distance from the wall. While there are many other factors which determine these model coefficients as mentioned by Sarghini et al. [36], for this thesis only the x location is considered. This allows to find different model coefficients for different x locations making the model more realistic. A schematic of the neural network in this case is given in figure 3.2. It can be seen that now there are 5 hidden neurons.

Flow within the network

Similar to the previous case, the equations for each layer of the network is given in this section. The output from the input neuron is then given by Z_{input} given in equation 3.1 where f is the activation function and b is the bias of the neuron.

$$Z_{input} = f((w_1 \times x) + b_{input}) \tag{3.3}$$

This output from the input neuron is taken as the input into the hidden neurons and the equation for one hidden neuron is given below. Here w_{11-23} is the weight of the connection between the first layer,first neuron to the second layer, third neuron.

$$(Z_{hidden})_3 = f((w_{11-23} \times Z_{input}) + b_{hidden1}) \tag{3.4}$$

Since there are several inputs to the output neurons, the product of the output of the hidden neuron and the weight of the connection between the hidden neuron and the output neuron must be summed up before applying the activation function. The result for each of the two output neurons is given by equation 3.5a and equation 3.5b respectively.

$$Z_{output1} = f(\sum_{i=1}^{5}(w_{2i-31} \times (Z_{hidden})_i) + b_{output1}) \tag{3.5a}$$

$$Z_{output2} = f(\sum_{i=1}^{5}(w_{2i-32} \times (Z_{hidden})_i) + b_{output2}) \tag{3.5b}$$

Similar to the previous case, the outputs are the required model coefficients. The result would give model coefficients for each x location, thereby making each simulation make use of several values of model coefficient depending on the x location. This is the forward pass of a neural network training. Before jumping into the backward pass, a training measure used in this ANN SGS model is studied.

3.2. Training

At this point, the layout of the neural network and the basic working is estabilished. Therefore it is shown how the network finds a value of model coefficients for a given x. However it needs to run a backward pass and several iterations to optimize these results to obtain the required results or the closest match. Usually in regression problems, the target is known and the neural network is used to fit the data to the target with minimum error. In that scenario, the error becomes the difference between the output of the neural network. This error is then required to be minimized. There are several ways to do this and these ways are termed as training algorithms.

The most common practice in neural networks by which this is done is the backpropogation. This method was first introduced by Rumelhart et al. [35] in 1986, where it is used along with an optimization technique,that is the gradient descent. This algorithm contains a two-phase cycle : the forward pass and the backward pass. In the forward pass, the input vector is passed through the network, layer by layer,using activation functions and initial weights and biases. The output of the layer is compared to a desired output and the error is calculated. The error is then propagated backwards, and the weights and biases are updated depending upon the contribution it has to the total error. The error derivatives are used to determine this. While basic backpropogation has its disadvantages such as slow learning convergence rate, the framework upon which it is based, is used with many faster algorithms to successfully and efficiently train a neural network. For the current study a Levenburg Marquadt Algorithm(LMA) is used. The reason behind this choice is the superiority of this method when moderate data sets and smaller networks are used which is perfect for the current state of the ANN SGS model.

3.2.1. Levenburg Marquadt Algorithm

With the training algorithm also set, one question remains. What is the target for the current case? The answer to this is a complex one. As there is no direct target for the model coefficients, there is a need for a performance term which the neural network could use as the error term. The performance term should use the output of the neural network and provide an estimate of how close the result using calculated model coefficients is from the DNS results. This performance term needs information from the fluid model to make the required estimate.

3.3. Performance Term

In terms of the fluid model, the model coefficients are used to calculate the small scale influence which is fed into the resolved scale equations. The resolved scale equation is then calculated for the required time as per the requirements of the user. [1]. The final velocity obtained from this LES calculation is then compared nodally to the DNS results are the LES nodes. The difference between the two results is used to find the simulation error measure(SEM) which is given by equation **??**

$$E_{SEM} = \sum (U_{les} - U_{dns})^2 \tag{3.6}$$

3.3.1. Simulation Error Measure(SEM) for training

Since the model is required to create results which are closer to the DNS values, the most accurate manner by which this could be done would be by using the SEM for training of the neural network. By doing this, the neural network would produce gamma's such that the SEM is the lowest, which ensures that the solution produced by this model is closest to the DNS values. However, the ultimate purpose of this model is to increase computational efficiency of an LES calculation without compromising too much on its accuracy. In order to use the SEM for training, the LES calculation would have to be completed for every iteration of the neural network which would make the model itself inefficient in terms of cost effectiveness. Therefore, another performance term is needed by the neural network which could be used like the SEM. For this purpose, a training error measure(TEM) was used initially by Durieux [10] which was used again by [3] for the creation of the current ANN-SGS model.

3.3.2. Training Error Measure(TEM) for training

The Training Error Measure as used in this thesis is given by equation 3.7. It is the averaged sum of the squares of the residual of weak form for each test function.

$$E_{TEM} = \frac{1}{N_t} \sum_{i=0}^{N_t} \sum_{j=0}^{N_w} (e_{i,j})^2 = \frac{1}{N_t} \sum_{i=0}^{N_t} \sum_{j=0}^{N^2} [(w_j, \mathscr{L}u - f)]^2 \tag{3.7}$$

Here e stands for the error per test function. Expanding this error, equation 3.8 can be written.

$$e_{i,j} = (\bar{w}, f) - (\bar{w}, \bar{u}_t) + \frac{1}{2}(\bar{w}_x, \bar{u}^2) + \frac{1}{2}(\bar{w}_x, \tau_{ANN}\mathscr{R}) - (\bar{w}_x, \nu \bar{u}_x) \tag{3.8}$$

In terms of the computations, the model uses integration points within each element. The contribution of each integration point are multiplied with an integration factor. These contributions are then added up to approximate the error per test function(equation 3.9b). The equations for each ip is given by equation 3.9a.

$$e_{ip} = (\bar{w}_{ip}, f_{ip}) - (\bar{w}_{ip}, \bar{u}_{ip,t}) + \frac{1}{2}(\bar{w}_{ip,x}, (\bar{u}_{ip})^2) + \frac{1}{2}(\bar{w}_{ip,x}, \tau_{ip,ANN}\mathscr{R}_{ip}) - (\bar{w}_{ip,x}, \nu \bar{u}_{ip,x}) \tag{3.9a}$$

[1] This time should also be the same as the DNS case used for training as well

$$e = \sum_{ip=1}^{N_{ip}} (f_{ip} \times e_{ip}) \tag{3.9b}$$

The TEM uses the DNS reference for calculation of the residual of the weak form. This results in a gain in the cost effectiveness of the model. However, besides this, the TEM needs to be comparable to the SEM in order to defend its use as a performance term for training the neural network. This means that the results found by using the TEM should be close to what the SEM would provide. Since the neural network would train to provide model coefficients that give the minimum of the TEM, it must be true that the minimum of the SEM is also achieved at similar model coefficients. In other words, the two error measures should have similar minima. For the TEM given by equation 3.7, this was true with most of the cases tested by Beekman [3]. However Beekman [3] also stated that there were regions where the TEM had a finite value but the SEM diverged. Morever for a case of periodic forcing at a reynolds number of 100, Beekman [3] showed that the minimum of the TEM is located in a region where the simulation diverged, further raising questions about the reliability of the TEM for all cases. The question about the reliability of the TEM is dealt with, in this thesis where more challenging cases are used which could help truly understand the limitations of the TEM.

3.4. Process flow of the Training

As mentioned before, since there arent direct targets to calculate the performance required for the training, a performance term is devised which is coest effective. In terms of the working of the model, this performance term works as a link to both the fluid model and the neural network. This is required because the neural network finds model coefficients several times which needs to be used by the fluid model in order to determine the residual of the weak form. The neural network needs the error term in order to compute the derivates of the error with respect to the network outputs which are used to update the weights and biases. The flowchart shown below explains the process flow in the training phase of a simulation.

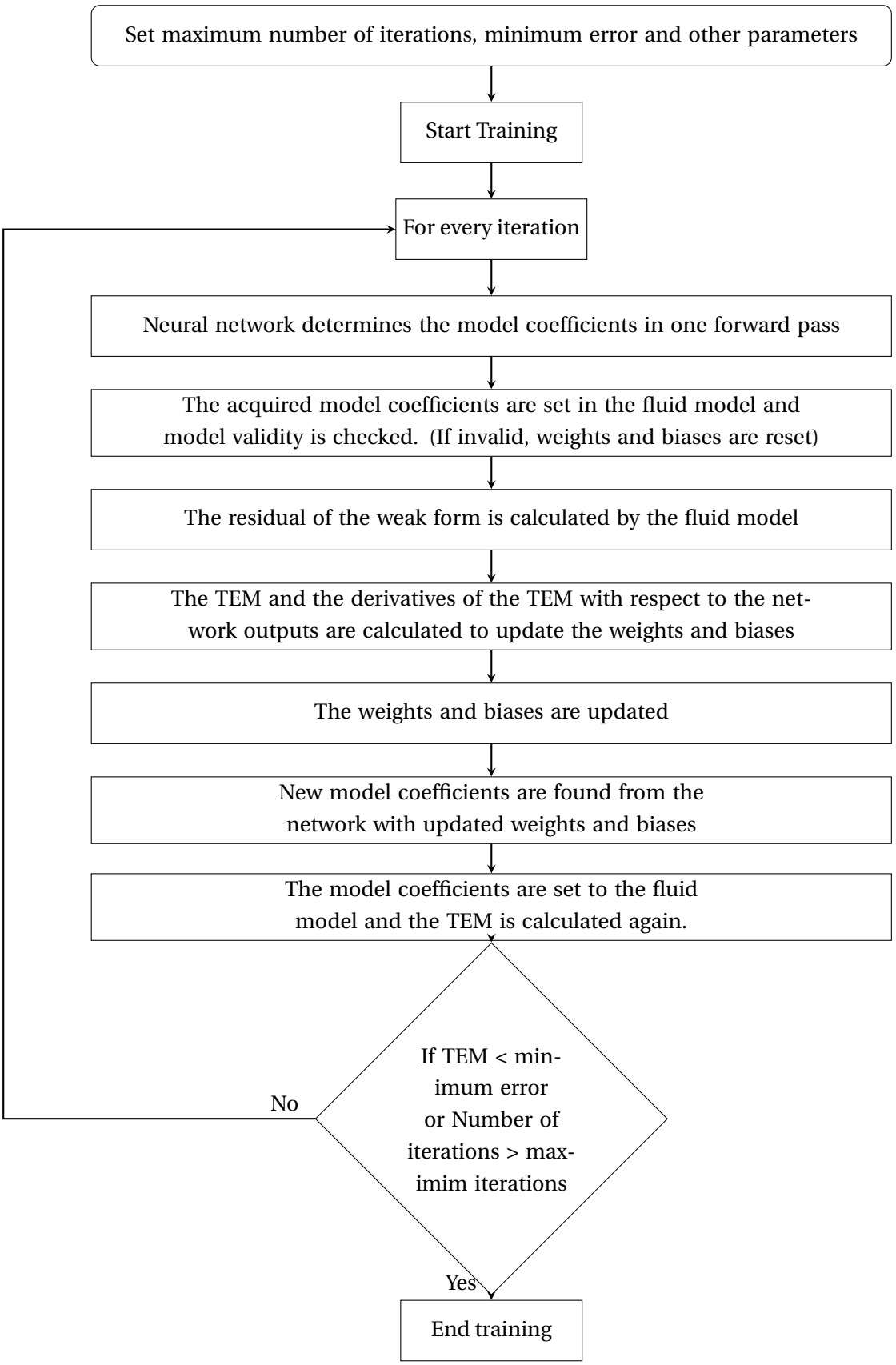


Figure 3.3: Process Flow Chart for Neural Network Training

3.4.1. Backward Pass

In the process flow it is explained that the model coefficients are updates using the derivatives of the error term. It would be wise to understand this backward pass with more detail, for the current case. The important result from the backward pass is the updating of the weights and biases. The training algorithm used is the Levenburg Marquadt Algorithm(LMA). It is an iterative technique that is used to find the minimum of a multivariate function. If f(x) is considered to be the output of the neural network, where x is the parameter vector(weights and biases) the following can be written through taylor's expansion for a small change δ_x

$$f(x+\delta_x) \approx f(x) + J\delta_x \tag{3.10}$$

where $J = \frac{\partial f(x)}{\partial x}$. Since it is a iterative method, several parameter values are calculated that minimize a particular quantity. In this case, the required δ_x for minimizing the quantity $||e - J\delta_x||$ is calculated at every step.The minimum is attained when $||J\delta_x - e||$ is orthogonal to the space of J[23]. This forms the following equations.

$$J^T(J\delta_x - e) = 0 \tag{3.11a}$$

$$J^T J\delta_x = J^T e \tag{3.11b}$$

The matrix formed by $J^T J$ is a matrix of second derivatives. In LMA, this diagonal elements of this matrix are valied from the original using a damping parameter. In other terms $N_{ii} = \mu + (J^T J)_{ii}$ where $N = (J^T J)_{LMA}$.

$$[J^T J + \mu I]\delta_x = J^T e \tag{3.12}$$

This damping parameter(μ) is increased throughout one iteration until an acceptable δ_x is found for the update of the parameters. The terms within the Jacobian are given by equation 3.13.

$$J(i,j) = \frac{\partial e_i}{\partial w_j} = \sum_{k=1}^{nIP} \frac{\partial e_{i,k}}{\partial w_j} \tag{3.13}$$

This means that the derivates of the error with respect to the parameters needs to be calculated. In order to do this, the derivatives of the error with respect to the outputs of the network is required. This is calculated through the following equations.

$$\frac{\partial e_{ip}}{\partial \gamma_1} = \frac{\partial e_{ip}}{\partial u'}\frac{\partial u'}{\partial \tau}\frac{\partial \tau}{\partial \gamma_1} \tag{3.14a}$$

$$\frac{\partial e_{ip}}{\partial \gamma_2} = \frac{\partial e_{ip}}{\partial u'}\frac{\partial u'}{\partial \tau}\frac{\partial \tau}{\partial \gamma_2} \tag{3.14b}$$

Equation 3.14 shows the different parts that are required for the calulation of the required derivative. The different parts of the equation 3.14 is given by equation 3.15.

$$\frac{\partial e_{ip}}{\partial \gamma_1} = \bar{w}_{ip,x}u' + \bar{w}_{ip,x}\bar{u}_{ip} \tag{3.15a}$$

$$\frac{\partial u'}{\partial \tau} = -\mathscr{R}_{ip} \tag{3.15b}$$

$$\frac{\partial \tau}{\partial \gamma_1} = -\frac{1}{2}\left[\gamma_1\left(\frac{u}{\Delta x}\right)^2 + \gamma_2\left(\frac{\nu}{\Delta x^2}\right)^2\right]^{-3/2}\left(\frac{u}{\Delta x}\right)^2 \tag{3.15c}$$

$$\frac{\partial \tau}{\partial \gamma_2} = -\frac{1}{2}\left[\gamma_1\left(\frac{u}{\Delta x}\right)^2 + \gamma_2\left(\frac{\nu}{\Delta x^2}\right)^2\right]^{-3/2}\left(\frac{\nu}{\Delta x^2}\right)^2 \tag{3.15d}$$

With the derivative with respect to the output, it is possible to get the derivatives of the hidden layer and the inner layes based on the layout of the neural network. For this the equations given in section 3.1 are used. Although the flowchart mentions only two possibilities due to which the training is stopped, there are mainly 4 causes.

- Error reaches a minimum value.
- Maximum Iterations are reached.
- The gradient of error i.e change in error value per iteration reaches a minimum value
- The difference between the updated weights and biases and the one of the last iteration is lower than a minimum value.

Besides all this, the current model also checks the validity of the model for the acquired model coefficients before and during training. The value of τ for some combinations of model coefficients would be invalid i.e a negative value would be obtained under the square root. To ensure the results are valid, before training while initializing the weights and biases, the validity is checked and the weights and biases are reinitialized if the model is invalid. During training, if this is encountered, the solution is rejected and the neural damping parameter is increased so that the step size is decreased. This ensures that the neural network works towards the required values soon.

4

Computational Setup

This chapter explains in detail the manner in which the model is built from a computational viewpoint. The chapters explains the libraries used in the model in brief and also verifies if it has been implemented correctly by using simple problems. Besides this, the different parts of the ANN SGS model is described in detail after which the changes made to the model are discussed. The verification of the renewed model is then presented. At the end, a flowchart of the new model is presented which explains clearly the information flow within the model during a simulation.

4.1. Libraries

The two main libraries used in this thesis was MEX and OpenNN. OpenNN is used for the neural network modeling and training whereas MEX is used for the fluid simulation.

4.1.1. MEX

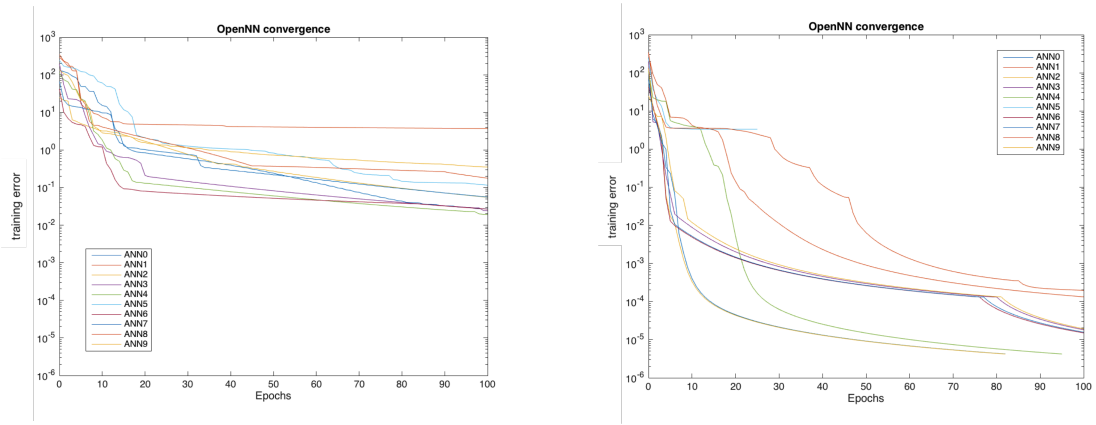
MEX(MFEM-extended) is a C++ library, created by Dr S.J.Hulshoff in the department of aerodynamics at the TU Delft, meant to solve transient partial differential equations using multiscale methods. It was built to serve as a extension for the MFEM which is an open source finite element model library. For the current variational problem at hand, this library can be used to make the model versatile and more computationally efficient.
The model as created by Beekman [3] used MEX for the fluid calculations. The current model uses a newer version of MEX (MEX v57) which allows easier integration with OpenNN, allows true 1D computations and also enables easy usage of different forms of the unresolved scales which would be beneficial for the fidelity test.

4.1.2. OpenNN

OpenNN is a library used for the construction and training of a neural network. This library was selected among various options by Beekman [3] because of the several characterestics it possesses. Firstly,t is an open source library making its accessibility easy. It is also a C++ based library which would simplify the icoupling of the two libraries significantly. It also provides the support to create an error class which is essential for the current problem, as there are no direct targets.

Beekman [3] made some changes to the basic openNN library. The LMA algorithm in OpenNN seems to use the gradient with respect to the total error instead of the individual error contribution. This makes the update equation as shown in equation 4.1

$$[J^T J + \mu I]\delta_x = 2J^T e \tag{4.1}$$



(a) OpenNN - Original

(b) OpenNN - Factor of two removed

Figure 4.1: Fitting a sin function with OpenNN(Beekman [3])

When compared to the update equation from theory as equation 3.12, it can be seen that an extra factor of two is present in the equation because the error contribution is taken with respect to the total error. This factor of two is then removed and a noticable increase in performance is observed. This is clearly shown from the results of Beekman [3] as shown in figure 4.1. Besides this openNN has also been changed to allow the use of a inbuilt performance term which also checks the output of the ANN and determines its validity with respect to the fluid simulation.

4.2. Model Build-up

The ANN-SGS model consists of various objects which contributes to a certain task by itself or with the assistance of other objects. This kind of a structure allows clear organisation based on role and saves on computational memory by allowing objects to access information relevant to their task alone. In this section, the different objects used in the ANN-based SGS model are described in detail.

4.2.1. Finite Element Model

This is the object which performs all the LES calculations and is a part of MEX. This object can take in custom values of model coefficients at every x location and use them for calculating the small scale influence. This object can also provide a validity check by ensuring the given values of model coefficients do not produce a negative tau argument. It works to also calculate the weak residual and provide that information to the performance term for every timestep.

4.2.2. Trainer

This is the neural network trainer. This object uses OpenNN extensively to create the neural network depending upon the architecture required. It is also responsible to set the scaling parameters, weights and biases (if it is hard set) and the training algorithm as well. This object trains the neural network and also provides the neural network with the link of the performance term which it can access while training.

4.2.3. Performance Term

This object calculates the final error term which the neural network tries to minimize by apadpting its weights and biases using the Levenburg Marquadt Algorithm. This object forms a link with the FEM model which provides this performance term with the weak residual for all the elements. This link is also useful when the performance term sets the model coefficients into the FEM model before calculating the TEM. The performance term then compiles these residuals for each timestep and

form the total residual which forms the training error measure. Moreover, the performance term also calculates the derivatives of the TEM with respect to the model coefficients which is used to make the Jacobian for the LMA. During the training, the validity of the model coefficients is checked by accessing the FEM model through this object. While the FEM model provides information at one timestep, the performance term is responsible for compiling and ensuring that the same is true for every timestep. The performance term is therefore the bridge between the FEM model and the neural network and also the learning part of the model which helps the neural network to optimize the model coefficients in the desired manner.

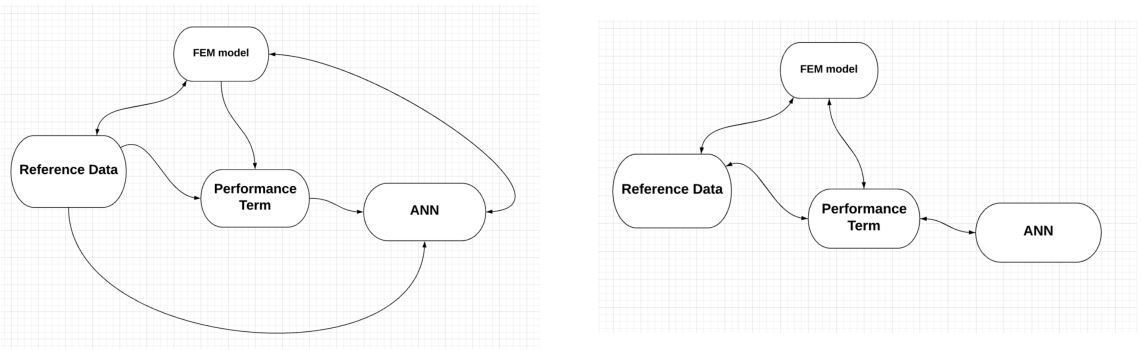
4.2.4. Reference Data

This object is used firstly as a database which provides information to the objects that need them. This is also responsible for creating the reference data from the DNS solution file. This reference data is also useful when creating response surfaces, to write the relevant data into the response surface creation files. This object is more of a data organizational object which helps to maintain order within the model.

4.3. Changes to the Model

For the purpose of the work carried out in the current thesis, it was required to make the model more modular so that different forms of the unresolved scales could be tested with the model. In order to do this several changes were made to the organisation of the model. Moreover, changes were also done to make the model more versatile.

4.3.1. Organization



(a) Unchanged Model

(b) Model with changes

Figure 4.2: Differences in the organization and information flow within the model

While the model as created by Beekman [3] worked well and was flexible to some extent, it did consist of several links which made it very difficult to change one object without causing changes in several other objects. This neccesitated the need to make the model more modular such that making changes could be easily done. From the computational perspective it made the model more modular and easier to handle. As it can be seen from figure 4.2 several links were removed from the new model and information flow is made cleaner. The ANN is the trainer which now works only with the performance term. All the objects are driven by a main code. This flow of information is easier to comprehend. As it can be seen the reference data provides information to both the performance term and the FEM model. This information is the relavant parametric details of the training set. Removing the link with the FEM model and the ANN helps in a way that now the model can be changed for different forms of the unresolved scale model with ease. It is clearly seen that even though the ANN uses the performance term, most of the information transfer happens between the

FEM model, the performance term and the Reference data.

Besides the organizational structure within the training model, changes have also been made to the layout of the complete model. While everything was done within the same model based on the parameter file, this process made the whole process a bit confusing. Therefore the different parts that for the complete model is now distinguished so that it is cleaner and easier to handle. The complete model involves :

- DNS model : This model creates the reference data for the training.
- Training model : This is the model which contains the neural network and uses it for training after which an LES simulation is carried out.
- Response Surface : This is a model that works in a way that a range of model coefficients and the step size is taken in, and each of the model coefficient pairs are fed into the model to perform an LES calculation. The results for every pair of model coefficients are stored in a file. Once all the model coefficients are checked, a script is used to generate a response surface to show the distribution of model coefficients and the corresponding TEM/SEM with the help of isolines. This is used to find out where the minimum of these error measures lie. While this isnt a part of the model, it is used to validate the performance of the neural network and the efficiency of the TEM.
- LES model: This is an optional model which could be used to generate the data that is currently obtained by an LES simulation. The goal is that the results provided by the training model must be better than the ones produced by this one.

4.3.2. Additional Changes

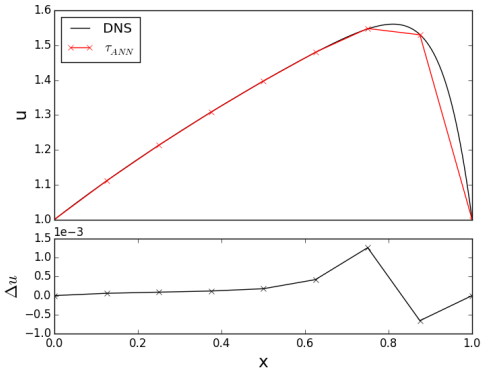
Besides simplifying the organisational structure of the model changes were made to make the model more versatile. The model now uses a parameter which is set before the simulation which would enable the creation of spatially varying model coefficients or spatially constant ones. Based on the parameter the neural network and its weights are modelled in the required way. A vector of model coefficients is created always to keep the spatially varying case consistent.The value of model coefficients during training is also saved into a file which could be used directly with the response surface.

4.4. Verification of the renewed Model

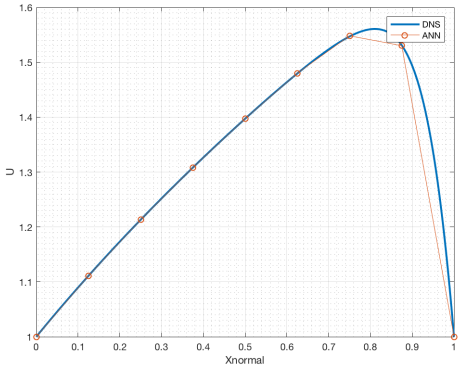
Since many changes were made to the structure and working of the model, it must be checked if the implementation is consistent with what was done by Beekman [3]. To check this, similar cases are considered and the results are compared. This result would bring verify that the current model works in a similar manner. Once this is done, a model is ready which works well and also is more modular thereby enabling the current author to perform different case studies to check its fidelity. This verification also ensures that the libraries are implemented correctly and working as intended. The two major cases considered in the research done by Beekman [3] were for both a reynolds number of 10 and 100. These cases was done for both steady and periodic forcing.

4.4.1. Steady forcing

The results for the case of reynolds number of 10 and 100 under steady forcing is given by figures 4.3 and 4.4. The results from the previous work is provided for comparison. As it can be seen, the two are the same.

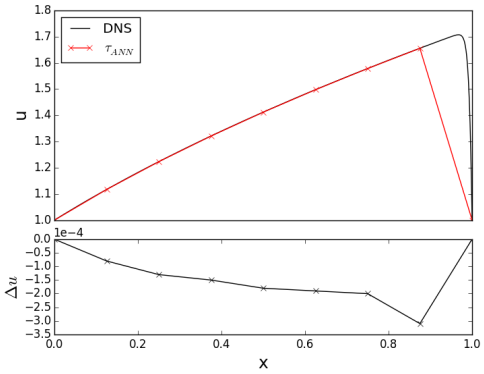


(a) Unchanged Model

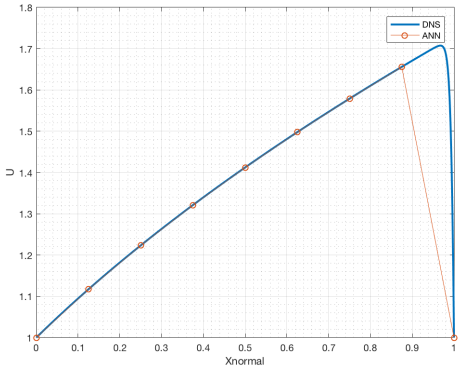


(b) Model with changes

Figure 4.3: Results for steady forcing at Re = 10

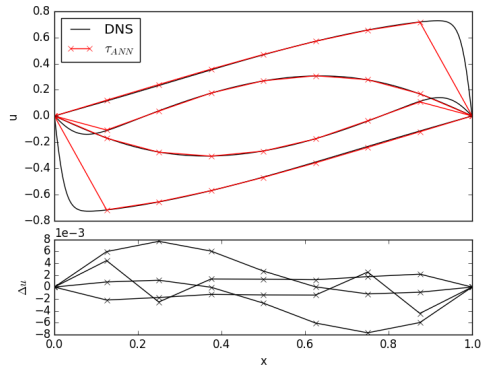


(a) Unchanged Model

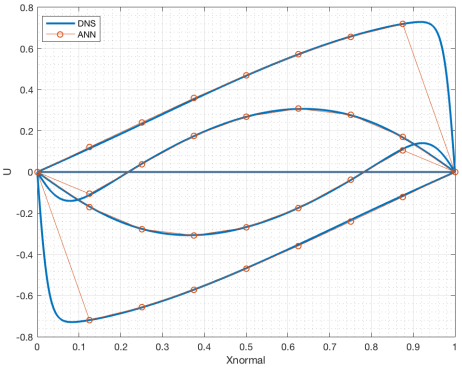


(b) Model with changes

Figure 4.4: Results for steady forcing at Re = 100



(a) Unchanged Model



(b) Model with changes

Figure 4.5: Results for periodic forcing at Re = 100

4.4.2. Periodic forcing

A similar comparison can be done for the case of periodic forcing as well. With this, it can be confirmed that the current model works well with the changes as well. The comparison for this case can be seen in figure 4.5. These results further affirm that the model is works as intended with the changes performed on it.

4.5. Flowchart of renewed Model

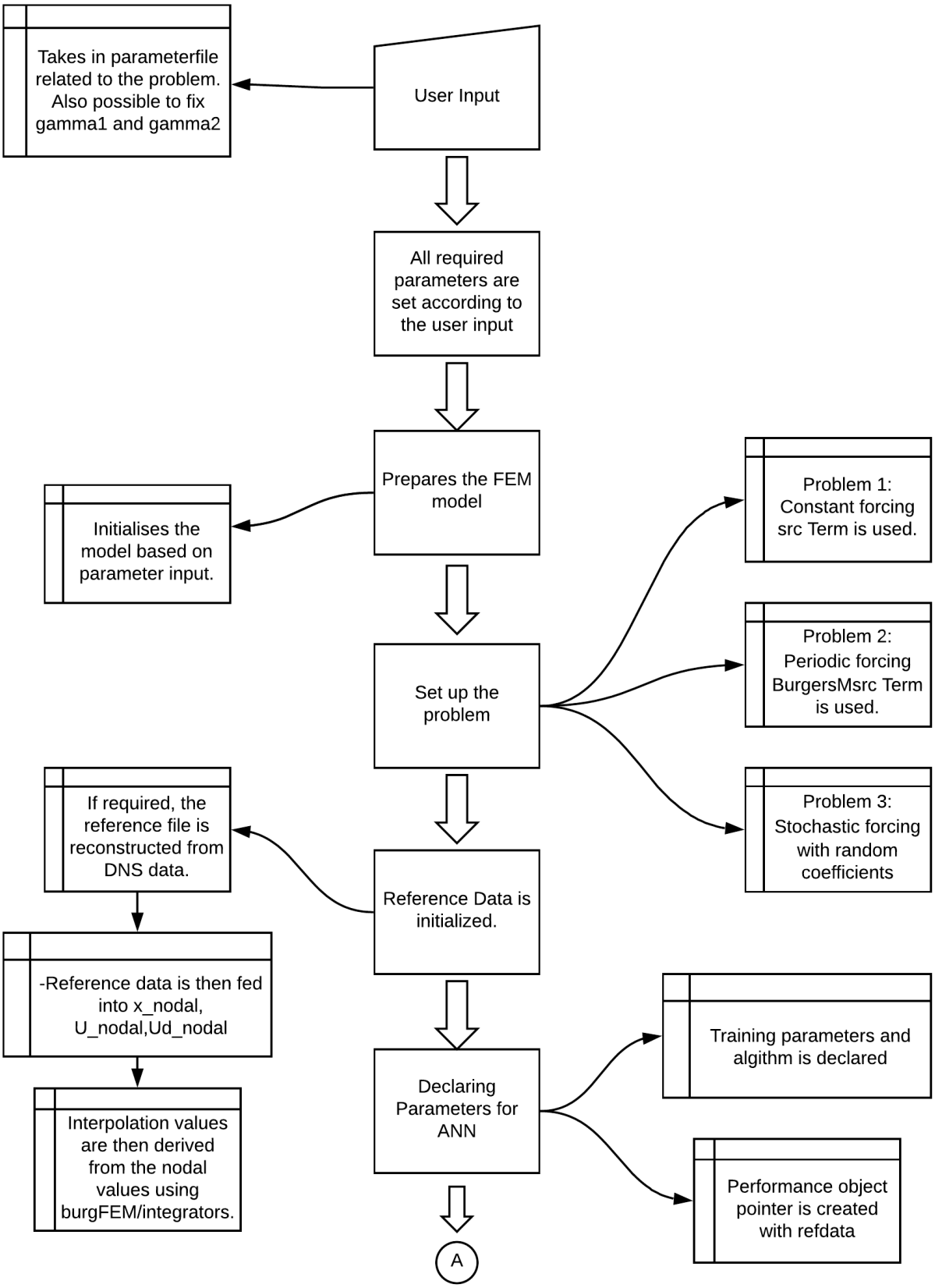


Figure 4.6: Flowchart for the current ANN-SGS model - Page 1

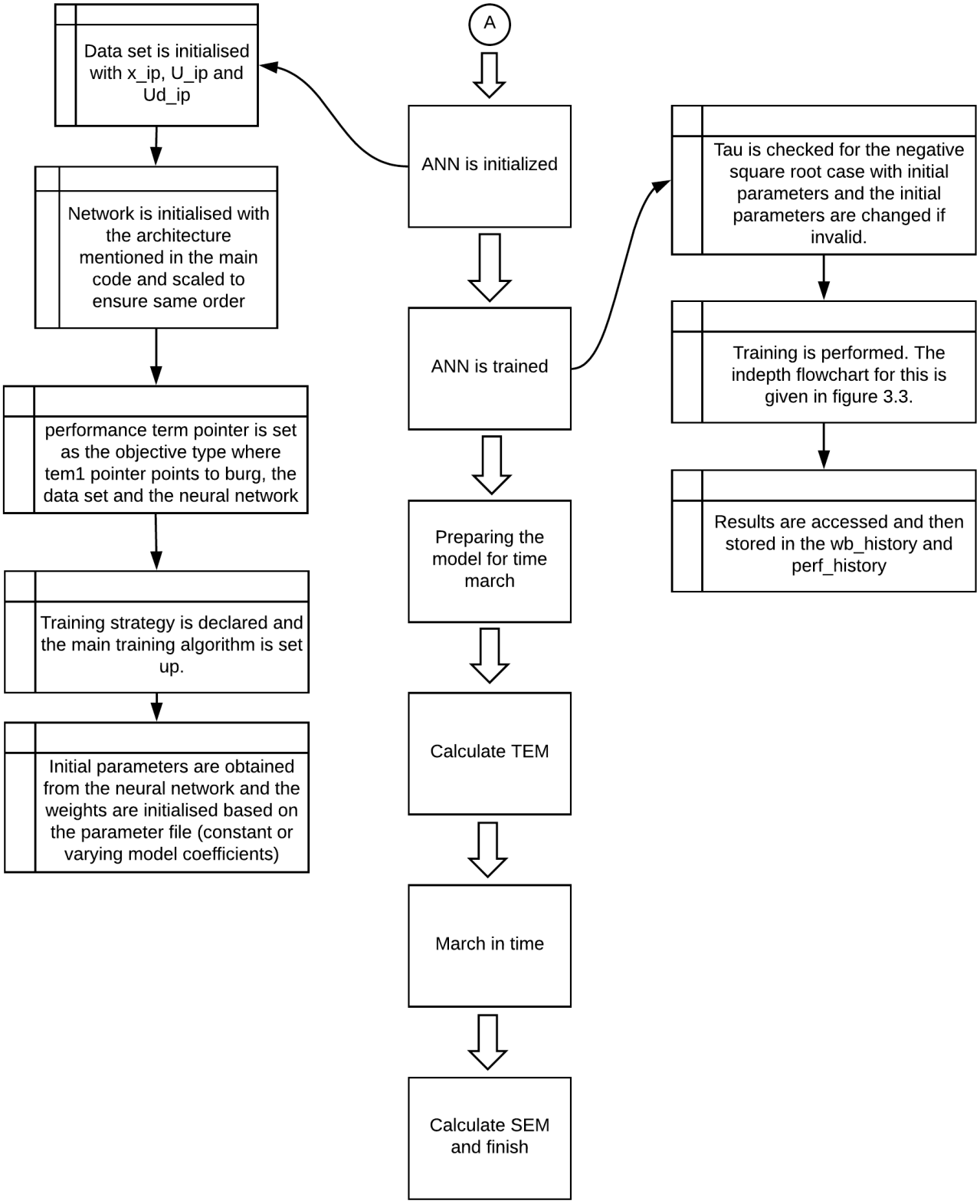


Figure 4.7: Flowchart for the current ANN-SGS model - Page 2

For the purpose of understanding and to help future colleagues working on the current model, a flowchart is given in figures 4.6 and 4.7. This flowchart explains the process in brief with reference to the several objects that were mentioned in this chapter previously.

5

Results

The previous chapter showed that the current model can reproduce the results from previous work. In this chapter the results obtained from the fidelity study are given. The analysis of the plots are performed and an explanation is provided. The results are obtained for three different forcing terms. All the problems are solved for the burgers equation given by equation 3.1. In all the cases, the Reynolds number was taken to be 10. Although a higher reynolds number can be used, it remains out of the scope of this thesis because of high computation times required for the DNS simulations. Since this thesis acts like a fidelity test for all the models, a reynolds number of 10 with different norms and forcing terms provide a deep understanding of how the ANN model works and its sensitivity to various parameters. For the Dynamic subgridscale(DSS) implementation, the implementation of the transient term is done on the orthogonal form of the unresolved scale model. Therefore all the DSS cases refers to this method unless explicitly mentioned otherwise.

$$u_t + uu_x - \nu u_{xx} = f(x,t) \tag{5.1}$$

The results are obtained for four different forcing terms ($f(x,t)$) and are given below.

5.1. Steady forcing

In this problem $f(x,t)$ is kept constant in both time and space ($f(x,t) = 1$). This problem is defined for the following intitial and coundary conditions.

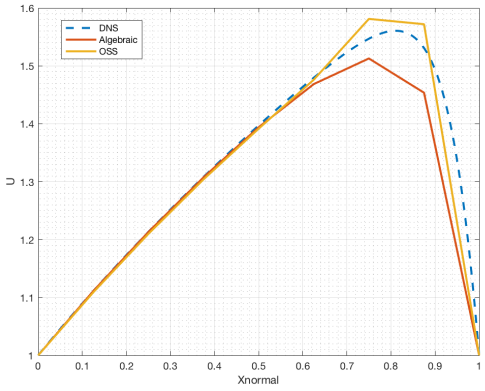
$$IC : u(x,0) = 1 \tag{5.2a}$$

$$BC : u(0,t) = 1; \quad u(1,t) = 1 \tag{5.2b}$$

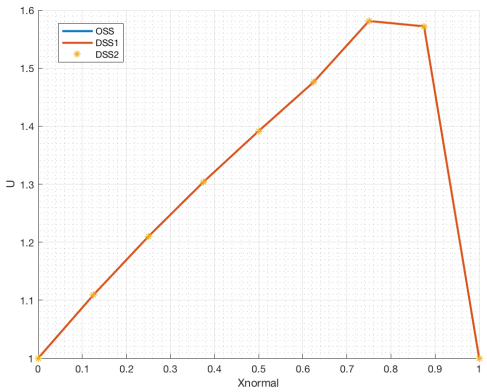
5.1.1. Standard Coefficients

In this subsection the different forms of the unresolved scale model are tested for a simulation time of 10 time units.

It can be seen from figure 5.1a that the algebraic(ASGS) model gives results that are not nodally exact.This is due to the fact that the model coefficients are not the best fit for a case like this one. It can also be seen that the orthogonal(OSS) form is very different from the algebraic one. In the research done by Guasch and Codina [13] as well as Ravichandran [34], it was suggested that the orthogonal subgrid scales is less robust than the algebraic model especially in the presence of sharp boundary layers which are not parallel to the velocity. Results from Ravichandran [34] suggests that there is a



(a) Solution for DNS,Algebraic SGS, and Orthogonal SGS



(b) Solution for Orthogonal SGS,DSS1 and DSS2

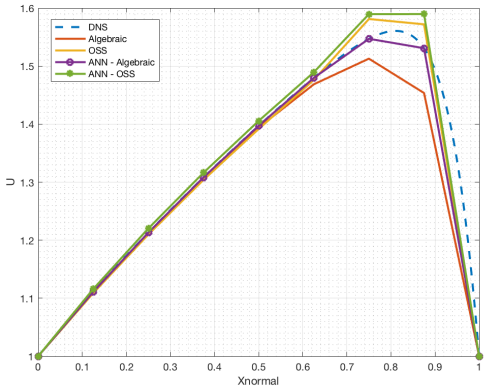
Figure 5.1: Comparison of different forms of the unresolved scale model for the classical case without the ANN

local overshoot for this problem when the orthogonal cases are considered. This forms as a validation for the orthogonal subgrid scales implementation.

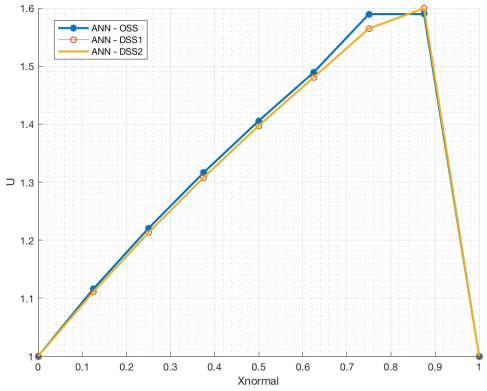
In the figure 5.1b it is also seen that both the DSS implementations provide results which are nodally exactly equal to the OSS results. This is expected because under a steady forcing, the effect of the transient term in the subgrid scale equations should be minimal.

5.1.2. Coefficients from ANN model

Now the ANN model results are checked in the similar manner as the previous subsection.



(a) Solution for DNS,Algebraic SGS, and Orthogonal SGS



(b) Solution for Orthogonal SGS,DSS1 and DSS2 in the ANN model

Figure 5.2: Comparison of different forms of the unresolved scale model for the classical case with the ANN

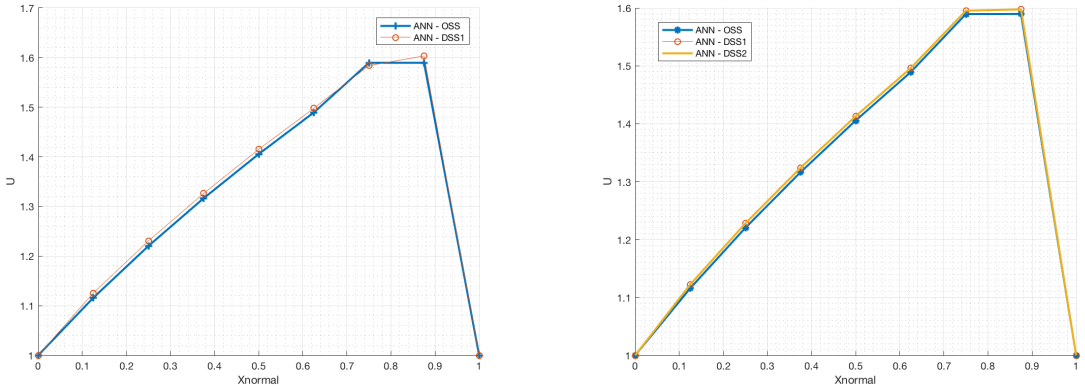
In the figure 5.2a it is seen that the results from the ANN (for the algebraic case) are nodally exact unlike the results found in the case without the ANN. This shows that for the steady forcing case the ANN produces model coefficients which are much better that the ones determined by Shakib and Hughes [38]. This further illustrates the results provided by Beekman [3] .

5.2a shows that for the OSS the ANN is unable to find model coefficients that fit the DNS nodal solution. This could be true because there might be no coefficients that fit to the projected DNS for this form of small scale influence for this case of steady forcing. Another interesting point is that

the solution is finds which corresponds to minimum training results in a solution that looks worse than the solution with the standard coefficients. In order to understand why this behaviour is seen, a deeper study is performed and the results are illustrated in the subsection 3.1.3.

5.2b shows that the DSS is different from the OSS. This should not be the case as the effect of the transient term must be minimum for the case of steady forcing.

5.1.3. Discrepency with DSS



(a) Solution for DSS with 20% transient training data (b) Solution for DSS with 40% transient training data

Figure 5.3: DSS training with transient DNS data

Since a discrepancy was seen in the DSS resuts, it needs to be understood. One of the reasons the DSS is so different is because in the figure 5.2b the training was done on steady DNS data. This means that the ANN trains to fit a steady data, however the effect of the DSS is pronounced only in the transient case. This causes the ANN to derive the model coefficients considering the DSS is nothing but an OSS essentially since there is no transient effect. However in the simulation, this effect becomes pronounced and the solution curves deviate as shown. To counter this, the ANN is fed with transient DNS data and the results are given in figures above. This confirms the necessity of providing transient data when DSS is used with the ANN-based SGS model.

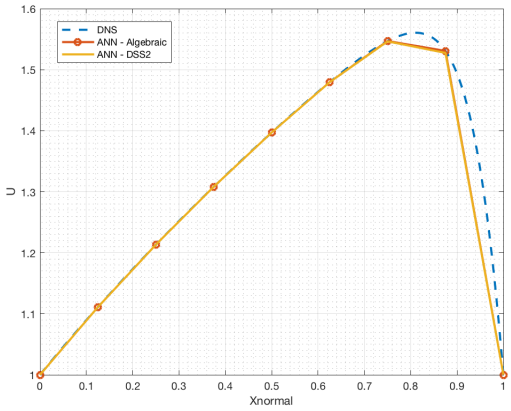


Figure 5.4: Solution of DSS used with Algebraic form of the unresolved scale model

From figures 5.3a and 5.3a it can be seen that the transient training data results in the DSS being closer to the OSS form of the unresolved scale. The effect of the amount of transient data can also be seen. If more transient data is available there is a better fit. In spite of the transient correction,

however there is a small difference between the DSS solution and the OSS solution. This can be attributed to the TEM deficiency with using OSS for this particular case, which will be explained in the next subsection.However to ensure that indeed, the DSS is not malfunctioning, the dynamic implementation was carried out with the algebraic form of the unresolved scale model which resulted in good correspondence as shown in figure 5.4. This shows that the DSS is indeed working correctly and the discrepancy with the OSS for this case must be further analysed.

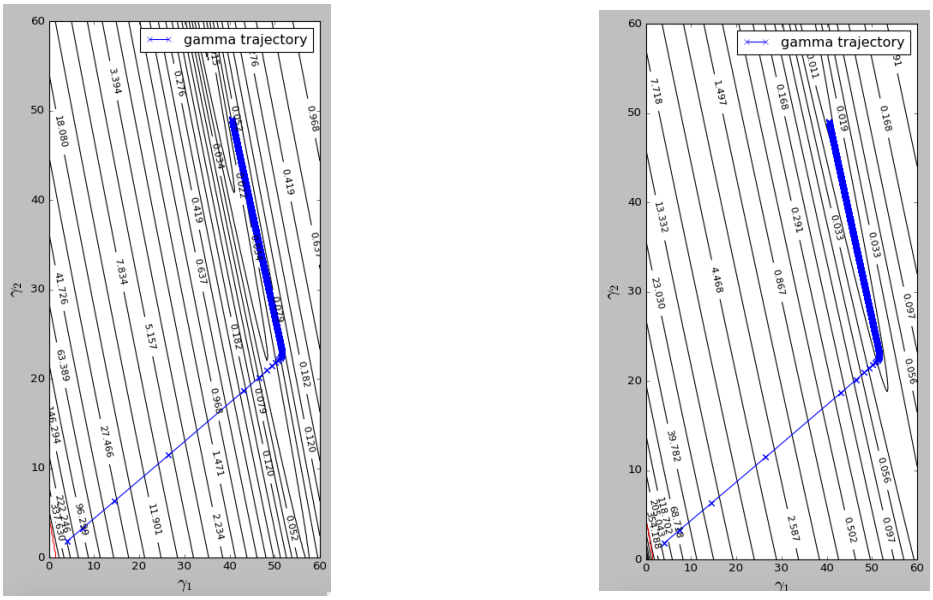
5.1.4. Discrepency with OSS

For all the cases considered up until now, a table is shown in table 5.1 with the Simulation Error Measure(SEM) and Training Error Measure(TEM) for both the standard coefficients and the coeffecients as determined by the ANN. It can clearly be seen that while the ANN produces a better TEM and SEM in both the algebraic and the algebraic with DSS, it only produces a better TEM for the OSS form of the unresolved scale model. The SEM is higher with the ANN trained coefficients. It is useful to plot the response surface for the OSS form of the unresolved scale model to see how the SEM and the TEM are effected by the model coefficients.

Table 5.1: SEM and TEM for the models with and without the ANN - Steady Forcing

	Standard Coefficients		**Coefficients extracted by ANN**	
	TEM	**SEM**	**TEM**	**SEM**
Algebraic	0.165118	0.274224	1.3028e-05	7.68449e-05
OSS	0.0404787	0.112328	0.0327654	0.211022
OSS - DSS1	0.0443993	0.11275	0.0380082	0.300003
OSS - DSS2	0.0443993	0.112751	0.0380082	0.299973
ALG - DSS2	0.0541915	0.272932	0.00569787	0.000479933

Response Surfaces



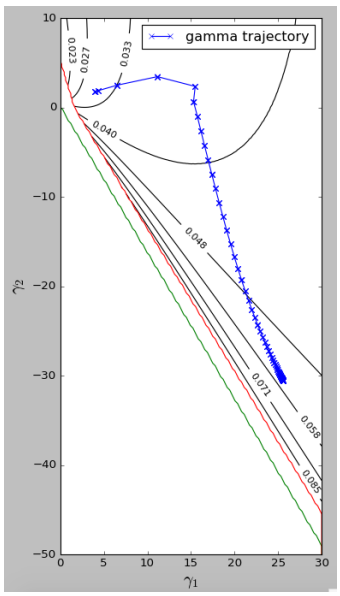
(a) SEM Response surface for steady forcing (Algebraic)

(b) TEM Response surface for steady forcing(Algebraic)

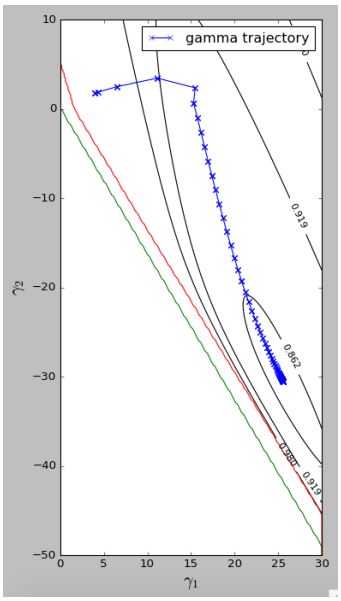
Figure 5.5: SEM/TEM Response surfaces for Algebraic

To understand the response surface and the desired effects, a response surface for the algebraic case is first shown. It can be seen in figure 5.5b that the ANN indeed trains towards the minimum TEM.

Another interesting fact as discussed by Beekman [3] is that the minimums of both the SEM and the TEM lie around the same model coefficients which makes the TEM useful for training.



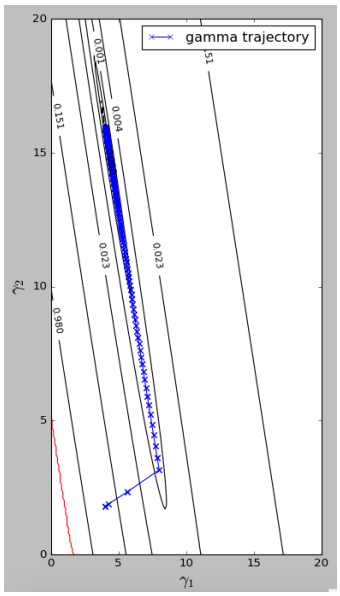
(a) SEM Response surface for steady forcing (OSS)



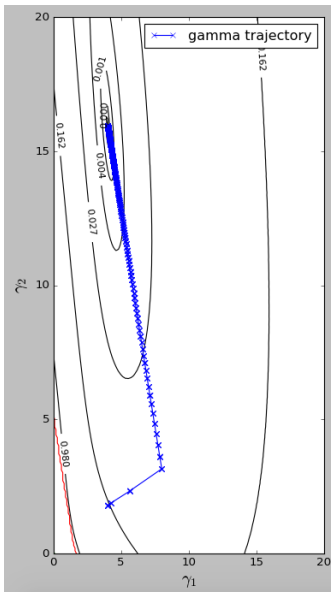
(b) TEM Response surface for steadyc forcing(OSS)

Figure 5.6: SEM/TEM Response surfaces for OSS

The response surface for the OSS form of the unresolved scale modelfor steady forcing is shown in figure 5.6a and 5.6b. From figure 5.6b it is clearly seen that the ANN trains to a minimum value of TEM. However the minimum of the TEM and the minimum of the SEM are far apart. There are two reasons why this could happen.One of them is that the ANN is unable to work with the OSS form of the unresolved scale model. The second is that the TEM is indeed deficient. Another interesting factor to the noted is that the SEM seems to have a low sensitivity to model coefficients.



(a) SEM Response surface for analysis of OSS



(b) TEM Response surface for analysis of OSS

Figure 5.7: SEM/TEM Response surfaces for OSS

Error Analysis

To isolate the error a response surface is drawn where the ANN is trained with the solution of the OSS form of the unresolved scale model with standard coefficients. If the ANN is able to train and find model coefficients equal to the standard coefficients, then it is able to work with the OSS form of the unresolved scale model and is able to find optimal model coefficients. This can be seen in figure 5.7a and 5.7a. Therefore it can be concluded that indeed there is a problem with the TEM. This could be true whether the form of the unresolved scale model is algebraic or OSS. It has worked so far as shown by Beekman [3] and Durieux [10] because the cases considered were simpler and always had model coefficients that could fit the DNS data very well. TEM still remains a very useful tool to determine the intial performance of the model however it would be worthwhile to consider using it along with SEM in such a manner that the performance is good and the computational effort is also low.

5.1.5. Spatially varying model coefficients

Since it has been seen that under the OSS form of the unresolved scale model, with the ANN training towards the TEM, the results worsen compared to the LES simulation with standard coefficients, it would be interesting to see if the model performs better if the model coefficients were to change in space. The results for both the algebraic and the orthogonal forms are given in sections below.A table is drawn showing the SEM and TEM values for comparison.

Table 5.2: SEM and TEM for the models with spatially constant and spatially varying model coefficients - Steady forcing

	Spatially constant Coefficients		**Spatially varying Model coefficients**	
	TEM	**SEM**	**TEM**	**SEM**
Algebraic	1.3028e-05	7.68449e-05	1.07073e-06	2.17594e-05
OSS	0.0327654	0.211022	0.01201	0.0476617

Algebraic Subgrid scale(ASGS)

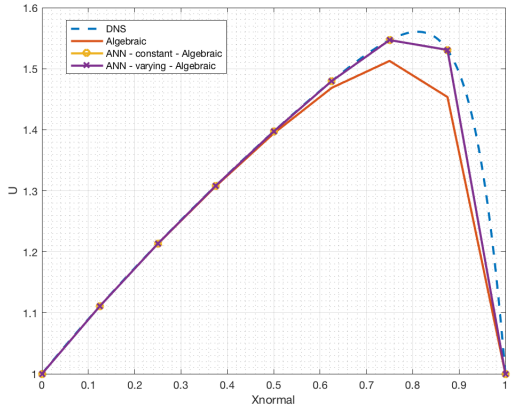
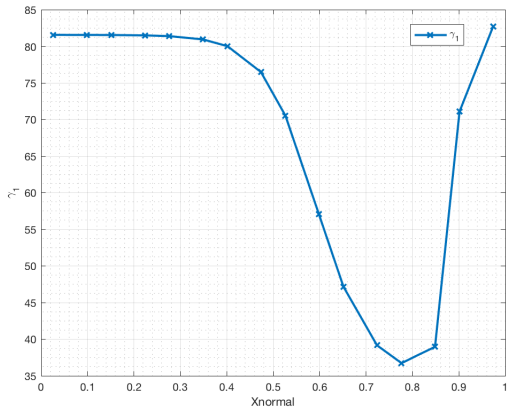
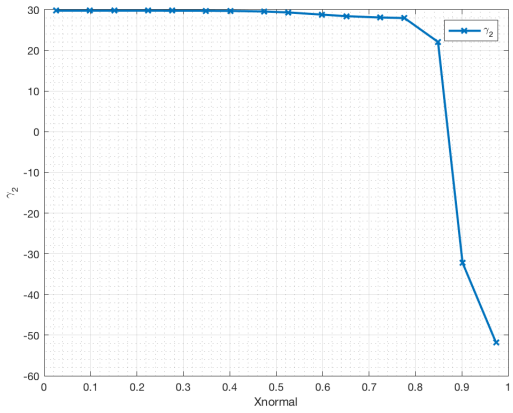


Figure 5.8: Spatially varying model coefficients with respect to other models - Algebraic case

From figure 5.8, it can be seen that the spatially varying case also produces a similar result like the spatially constant case. However, table 5.2 shows that the spatial case indeed provides a better SEM and TEM. At this point it would be worthwhile to see the distribution of the model coefficients in space. This is seen in figure 5.9a and figure 5.9b. Compared to the constant standard coefficients of 4 and 16, this distribution is shown to vary quite a lot.



(a) Distribution of γ_2



(b) Distribution of γ_2

Figure 5.9: Distribution of Model Coefficients - Algebraic case

Orthogonal Subgrid scale (OSS)

The effect of spatially varying the coefficients is more pronounced in this case. Since an anomaly was found with the TEM and SEM in this case, with the constant coefficients, it is interesting to see how the results change if the coefficients were trained as a function of space. This is seen in figure 5.10. The results are much better, even exceeding the performance of the classical LES with constant coefficients of 4 and 16. This shows the true effect of the neural network, where the model coefficients are determined as a function of space. This is a more realistic model and hence the results form closer to the reference. However the question still remains if the TEM is indeed the best measure for challenging cases.

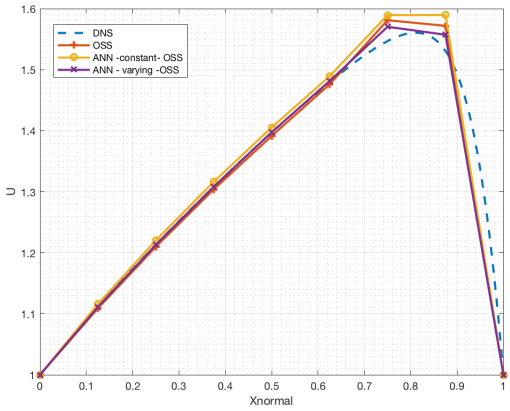
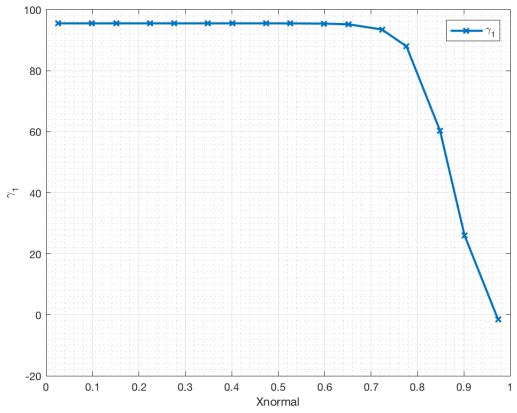
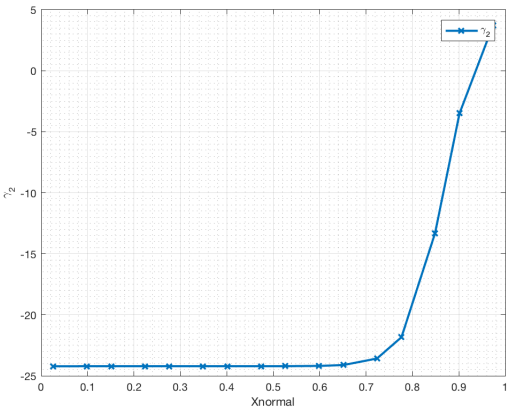


Figure 5.10: Spatially varying model coefficients with respect to other models - Orthogonal case

Like the algebraic case, a distribution of model coefficients for the OSS case is shown in figure 5.11a and figure 5.11b. The model are again seen to vary over a large range of values.



(a) Distribution of γ_2



(b) Distribution of γ_2

Figure 5.11: Distribution of Model Coefficients - OSS case

5.2. Periodic forcing

In this problem $f(x,t)$ is varied in both time and space defined by the equation 3.3. The same case was tested by both Beekman [3] and Durieux [10]. The LES calculations have been done for a total of 8 number of elements, whereas the DNS was done for 1024 elements. This is the case for both steady and periodic forcing

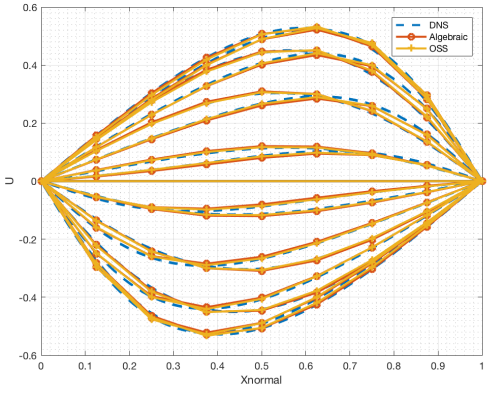
$$f(x,t) = \sin(\frac{\pi}{2}t) \cdot \sin(\pi x) \tag{5.3}$$

This problem is defined for the following intitial and coundary conditions.

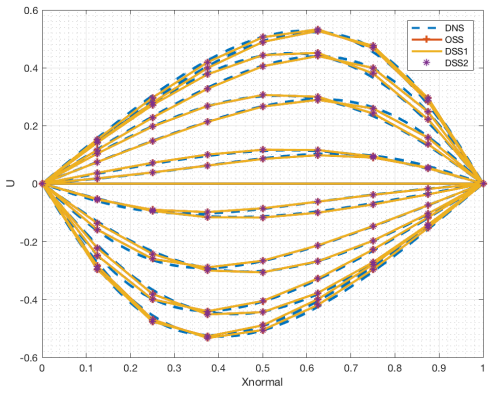
$$IC : u(x,0) = 1 \tag{5.4a}$$

$$BC : u(0,t) = 1; \quad u(1,t) = 1 \tag{5.4b}$$

5.2.1. Standard Coeffients



(a) Solution for DNS,Algebraic SGS, and Orthogonal SGS



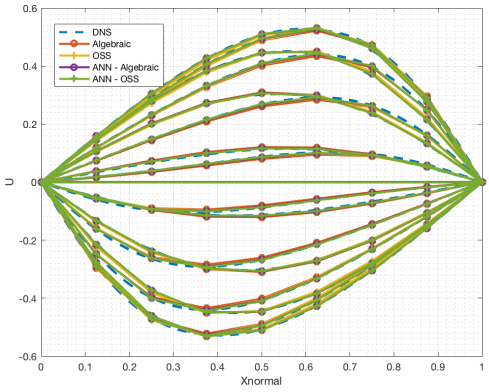
(b) Solution for Orthogonal SGS,DSS1 and DSS2

Figure 5.12: Comparison of different forms of the unresolved scale model for the classical case without the ANN

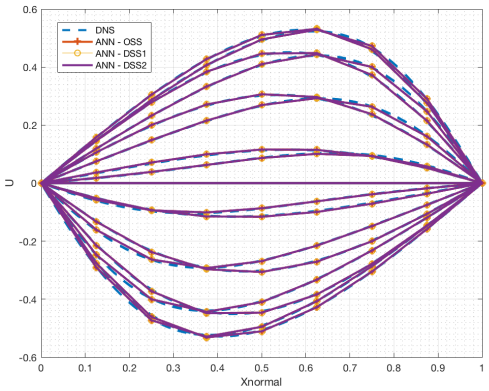
From figure 5.12a it can already be seen that for the case of periodic forcing the ASGS and OSS gives good results. For the DSS cases, both DSS1 and DSS2 is compared with OSS in figure 5.12b. It is clear that the all the three provide the same results. The local overshoot (for OSS) as seen in the case of steady forcing dissapears since the sharp boundary layer is not formed in this case.

ANN Model

A similar study is then done for the ANN based SGS model.



(a) Solution for DNS,Algebraic SGS, and Orthogonal SGS



(b) Solution for Orthogonal SGS,DSS1 and DSS2

Figure 5.13: Comparison of different forms of the unresolved scale model for the classical case with the ANN

For periodic forcing the ANN model also provides good results for the OSS as well. This is clearly seen in figure 5.13a where the results from the model without the ANN and with the ANN for the OSS form is almost indistinguishable.

However figure 5.13b shows no difference between the DSS and OSS further proving that the DSS works well with the ANN if given enough transient training data.

Since both the standard coefficients and the coefficients produced by the ANN produce solutions that look similar, it would be useful to refer to table 5.3 to see the SEM. It is seen that the ANN produces much smaller SEM in all the cases.

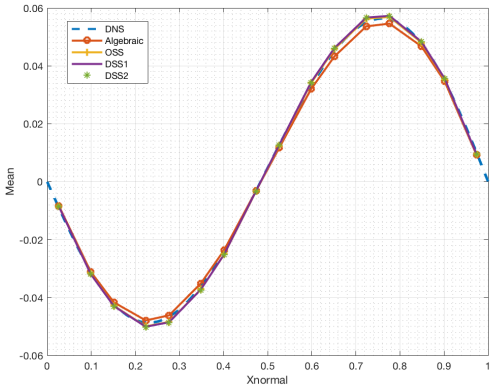
Table 5.3: SEM and TEM for the models with and without the ANN - Periodic Forcing

	Standard Coefficients		**Coefficients extracted by ANN**	
	TEM	**SEM**	**TEM**	**SEM**
Algebraic	0.000299947	0.00800793	4.90354e-05	0.00280713
OSS	0.000505063	0.00861921	8.91542e-05	0.00265377
OSS - DSS1	0.000282387	0.0092415	8.49997e-05	0.00266052
OSS - DSS2	0.000282387	0.0091251	8.52019e-05	0.00266043

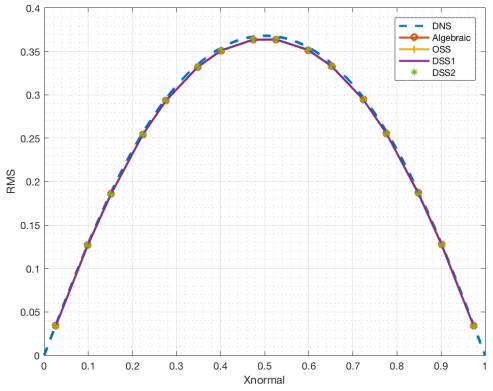
5.2.2. Mean & RMS

Since an instantaneous solution only provides the solution at a particular time and since the forcing is variant in time and space, it is beneficial to plot the time averaged mean. Figure 5.14a shows the time averaged mean which shows that the algebraic and OSS forms produce slightly different means. Figure 5.15a shows that the ANN produces the same mean for all forms but it is a bit different from the DNS mean.

RMS plot can also be seen in figure 5.14b which further shows that the standard coefficient results are very close to the DNS case. Figure 5.14b shows that the ANN produces an almost exact RMS plot as the DNS plot.

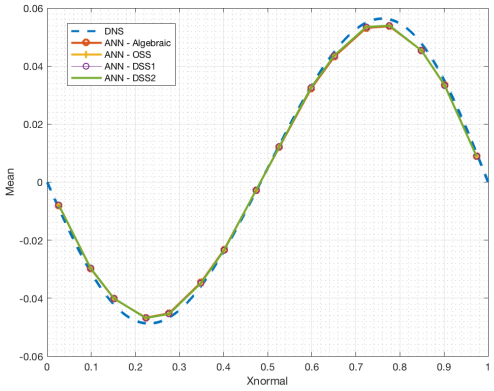


(a) Mean for periodic forcing - Standard coefficients

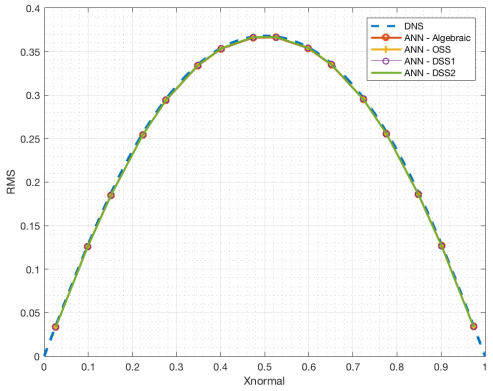


(b) RMS for periodic forcing - Standard coefficients

Figure 5.14: Mean and RMS plots for periodic forcing - Standard coefficients



(a) Mean for periodic forcing - ANN model



(b) RMS for periodic forcing - ANN model

Figure 5.15: Mean and RMS plots for periodic forcing - ANN model

5.3. Stochastic Forcing

Since the periodic forcing works well for the OSS form of the unresolved scale model, it would be nice to further complicate the problem and see the effect of the same. A forcing function as defined by Chambers et al. [4]. The construction of the forcing term was defined in Chambers [5] which is adapted for the purpose of this study. The forcing term is defined by the following equation.

$$f(x,t) = \sum_{k=0}^{K} (a_1(t)\cos(2\pi x) + a_2(t)\sin(2\pi x)) \tag{5.5}$$

where the coefficients $a_1(t)$ and $a_2(t)$ is found for every forcing wavenumber (k) as a function of random numbers between 0 and 1, Reynolds number and timestep. This forcing is then found for every wavenumber defined. In this case forcing wavenumbers used are 1 to 41. For every time step these random numbers are generated again to find new coefficients and the forcing term is defined again. This forcing is interesting because the results of a computation with this type of forcing is shown to mimic the pressure term which is negated in formation of the burgers equation from the Navier stokes. Therefore the results could essentialy give a first idea of how the results on a 1D Navier Stokes would look like. This problem is defined for the following intitial and coundary conditions.

$$IC : u(x,0) = 1 \tag{5.6a}$$

$$BC : u(0,t) = 1; \quad u(1,t) = 1 \tag{5.6b}$$

5.3.1. Validation

Before using the stochastic forcing for testinf the ANN model, it must be checked if the implementation is in accordance with the one shown in Chambers et al. [4]. In order to do this the problem is run for a 10000 time units (This is to ensure the solution has converged).The results shown in Chambers et al. [4] are for reynolds number of 413.5 and above. Since a validation cannot be performed for high reynolds number due to limitation of time, a qualitative validation is done to ensure that the implementation is fairly accurate.

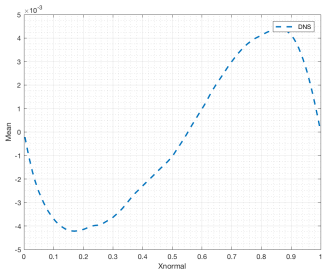


Figure 5.16: Time averaged mean for Re 10

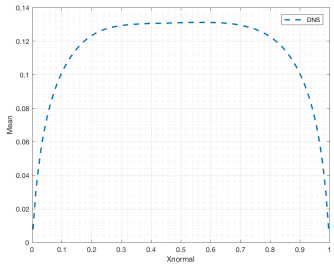


Figure 5.17: RMS plot for Re 10

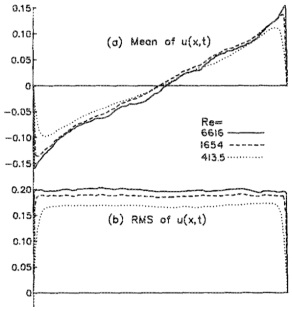
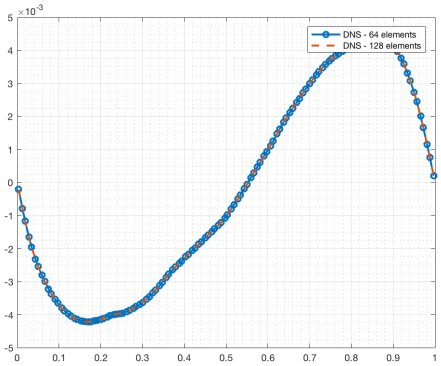


Figure 5.18: Reference plots (Chambers et al. [4])

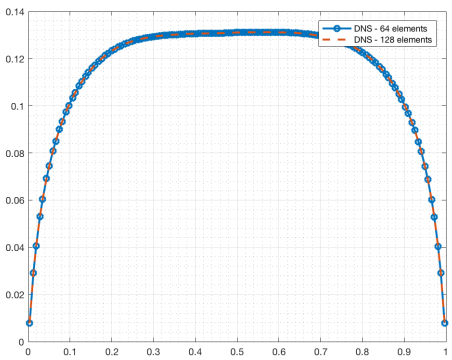
It can be seen that the mean profile in figure 5.16 and the RMS plot in figure 5.17 looks close to the ones shown in reference (figure 5.18). It can be seen in the reference that the ends start to curve for lower reynolds number which is seen in the authors case of Re 10. One feature that can be seen is the small waves in the mean plot. But this is also seen in the reference case and can be attributed to the effect of this forcing term.

For the purpose of this thesis, only a qualitative validation is shown. Since the reference shows results for higher reynolds number, a lot of computational effort is required to completely validate this forcing term. This was done in the MSc Thesis of Donnatella Xavier. The readers are encouraged to turn to this work for a more refined validation study.

5.3.2. Grid dependency



(a) Time averaged mean



(b) RMS plot

Figure 5.19: Comparison plots for n = 128 and n = 64

In order to check the optimum number of elements for this forcing term, the case is tested for both 64 elements as well as 128 elements. The results show that the effects are in fact fully captured even

for a grid of 64 elements and hence this is taken as the DNS result for the case of inhomogenous stochastic forcing.

5.3.3. Effect of Simulation time

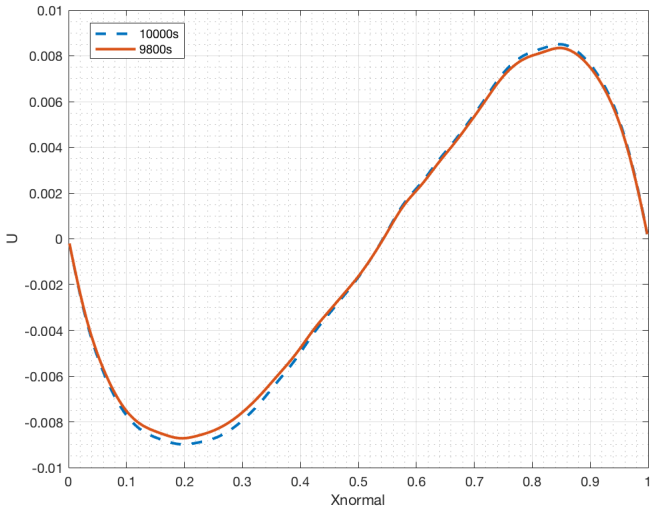
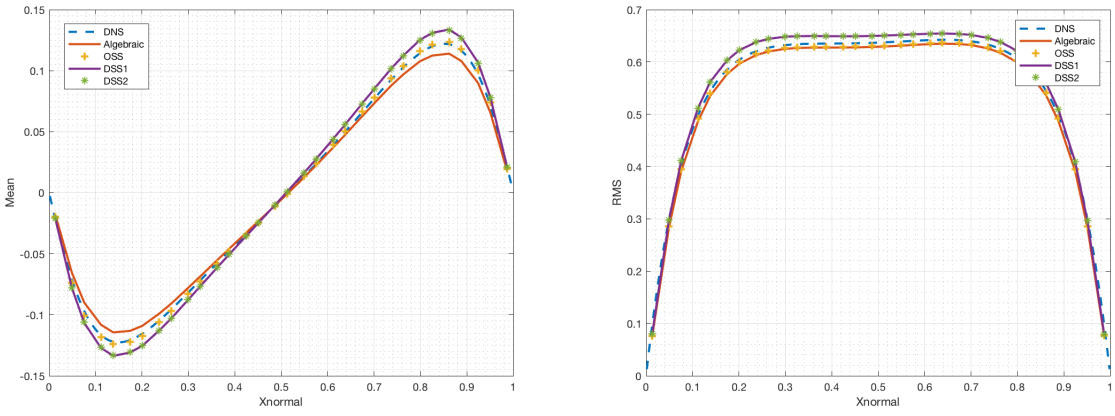


Figure 5.20: Results at different simulation times

The solutions at 9800 time units and 10000 time units seems to be similar with slight differences. However for the anaysis a simulation time of 10000 secs is used.

5.3.4. Mean & RMS

Figure 5.21a shows the time averaged mean of all the LES calculations without the ANN as well as the DNS result for reference.The mean profiles were calculated after a time of 10000 time units to ensure that the initial transient effects are ignored in the mean profiles. The figure 5.21b shows the RMS plot of the calculations.



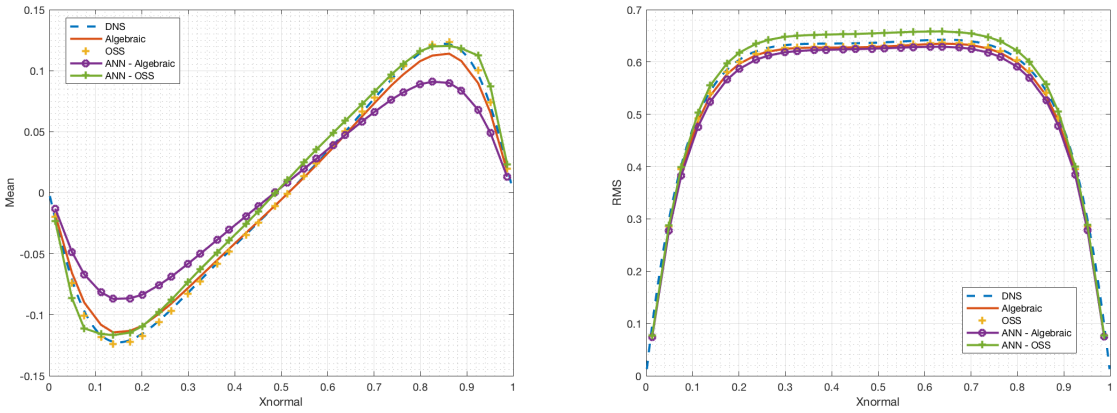
(a) Time averaged mean

(b) RMS plot

Figure 5.21: Comparison plots of Mean and RMS for long simulation time

It can be seen from figure 5.21a that the OSS values already produce a good result without the ANN. The algebraic form of the unresolved scale model however is unable to produce such a good result. The dynamic forms also seem to provide results that aren't nodally exact. The reason the DSS does

not provide as good results might be that the transient training data fed in to the ANN is only 4% of the total data. This is because the simulation is done for a very large time producing impractical amounts of data. For the purpose of this study, for this particular case, more forcus is put towards the OSS and algebraic forms. The figure 5.21b shows that the OSS form although fits the mean profile quite well, the RMS values still vary a little.



(a) Time averaged mean

(b) RMS plot

Figure 5.22: Comparison plots of Mean and RMS for long simulation time

From the mean profiles for the case with the ANN as seen in figure 5.22a, it is clearly visible that the ANN is unable to provide a better result than LES simulation without the ANN. Also, the algebraic form seems to be quite different than what is expected. In order to give more insight on the working of the ANN for this case, a table is given below with the SEM and TEM values(table 5.4). It can however be seen from the RMS plot in figure 5.21b that the error in the RMS is not much affected by the presence or absence of the ANN.

Table 5.4: SEM and TEM for the models with and without the ANN - Stochastic forcing

	Standard Coefficients		**Coefficients extracted by ANN**	
	TEM	**SEM**	**TEM**	**SEM**
Algebraic	37199	0.00083732	17940.9	0.01693185
OSS	36339	0.00009747	17724.1	0.001730378

From the table 5.4 it can be seen that the TEM is smaller for all the cases with the ANN, which means that the ANN is working effectively. However, the SEM is much higher. Here the SEM is the error between the mean values of the simulation mentioned with respect to the mean values of the DNS simulation. The error in the results of the ANN could be attributed to the deficiency of the TEM, since the TEM is actually lower for both the forms. Another source of error could be the limited sampling data that is used for training. Moreover the forcing wavenumbers used in this case are over a large range which makes the case very realistic, however this might cause the forcing to interact with the small scale influence directly, making the SGS model more complicated.

6

Conclusions

In this chapter, the conclusions are drawn from the analysis given in chapter 5. In this study, an ANN-based SGS model was remodelled and used to perform several fidelity tests to understand the range of application of the procedure developed for constructing ANN-based SGS models. This chapter revisits the research questions and answers them based on the results.

- *Does the ANN based SGS model work well even for cases with high nonlinearity?*

  The Stochastic forcing test case provided results which were close to the reference but were not better than the results from a OSS with standard coefficients. The reasons were speculated to be multifold. One of the key reasons, is the deficiency of the TEM. It was found from the current study that the TEM, while serves as a qualitative initial estimate, would not be the best measure especially when model coefficients which fit the reference data do not exist. Another source of error could the limited time over which the network is trained. Lastly, the forcing was done over a large range of wavenumbers which although more physical, this may have intersected directly with u' making the interpretation of its behavious more complicated.

- *Does changing the model affect the accuracy and training method required?*

  It was shown from this study, that the general approach of the ANN based LES calculation is capable of working with different forms of the unresolved scale model. However it was also seen that the model seems to be deficient if there exist no model coefficients which fit the reference case, like the OSS for the steady forcing. In such a scenario, a modified performance term would be a better measure for training.

- *Does the ANN based SGS model work better than the LES with standard coefficients under different conditions? If not what could the reasons be?*

  It is shown that the ANN-SGS model does in fact work better than the LES with standard coefficients for all the forms under periodic forcing and for the algebraic SGS model under steady forcing. The stochastic forcing test case and the OSS model under seady forcing however, gave a different result. The main reason for this is the deficiency of the TEM.

- *Is the Training error norm effective for all the cases considered? If not,what changes need to be made to make it more universally correct?*

  No, the training error norm (TEM) is in fact deficient for cases when there are no any model coefficients that fit the projected DNS solution. In such a case the minimum of the TEM seems to be very different from the SEM, making the TEM incompetent for the purpose of training. This means that while using the current TEM it must be carefully decided which combination of projector and form of the unresolved scale model to use such that model coefficients exist which create a solution similar to the DNS projected solution. However for complicated cases like the 3D Navier stokes it is impossible to find exact solutions with LES. Therefore, while the TEM is a good meaure for initial training, the TEM needs to be altered in a manner such that the minimums of the TEM and SEM are around the same model coefficients. Another solution coud be to consider a different projector of the DNS solution such that coefficients to minimize the error do exist.

- *Would the effect of spatially varying coefficients be large or small?*

  The effect of spatially varying coefficients was shown to be significant. While constant coefficient OSS model with the ANN produced inaccurate results owing to the deficiency in the TEM, with spatially varying coefficients, the results were shown to be much better. Moreover, the distribution of model coefficients was found to be over a wide range showing that this spatial variance is indeed essential for more realistic simulations.

7

Recommendations

1. To counter the TEM deficiency, an attempt could be made by changing the projector of the DNS solution in a way such that the form of the unresolved scales would have coefficients that could result in the desired solution. For example, in the case of OSS, the L2 norm could be used in place of the nodal norm for the simulation error measure.

2. The TEM could also be used along with SEM in a manner that the computational effort is not compromised by much while finding better coefficients.

3. Another noticable factor was that the model seems to have low sensitivity to model coefficients when there exists no model coefficients to produce an exact solution. In such a case, it might also be worthwhile to complicate the neural network and allow it to train for τ directly instead of the model coefficients alone.

4. For the case of stochastic forcing, it would be beneficial to check if the TEM is a function of the randomness of the forcing function, and if it is, it must be ensured that this is not the case by taking relevant measures. Maybe a case of lesser randomness(forcing over a smaller range of wavenumbers or lesser amplitude) with lower convergence time could be studied extensively to ensure that the TEM works as just a function of the model coefficients.

A

Introduction to Turbulence

A.1. Wall-bounded Turbulence

Wall bounded turbulent flows are studied at different Reynolds number(Re) depending on its application. The knowledge of wall bounded turbulent flows at high reynolds numbers (10^5-10^8) is of great practical importance to the aviation industry. For example, in order to determine the skin friction coefficients effectively, it is necessary to study the flow near the surface of the wing. Another application would be to reduce the noise that occurs from the turbulent structures in the boundary layer.

As Re increases, the smallest scales of turbulence decreases by a factor greater than 175, thus causing severe challenges in the study of these flows (Metzger and Klewicki [27]) .These small scales of turbulence poses challenges for the spatial resolution of the sensor in experimentation thus necessitating the need for alternate techniques. (Metzger and Klewicki [27]).

Numerically these flows(upto Re = 5000) can be solved with good accuracy but with high amount of computational time and cost. This is owing to the fine mesh size in the near wall region. Piomelli [32] shows that at high reynolds numbers the computational cost is more than 50 percent for the inner layer alone. It should also be noted that most technologically significant flows, especially the ones in the atmosphere, are at a high reynolds number (Re>100000). This warrants the need for new models that can compute challenging turbulent flows at high reynolds number more efficiently. Research on this area has been increasingly active in recent years (Marusic et al. [24], Meneveau and Katz [26]) . In this chapter, section 2.2 further explains the current reseach in this area.

A.1.1. The Turbulent Boundary Layer

The boudary layer is considered to be composed primarily of two regions namely

- Inner layer : This is the region where the viscosity effects are considered. In this layer, the length scale is dependent on the viscosity of the medium and is defined by υ/U_τ where $U_\tau = \tau/\rho$.
- Outer layer : In this region, the viscosity effects are considered to be negligible. The length scale is equivalent to the boundary layer thickness δ.

However the underlying assumption is that the velocity scale for both the regions is considered to be U_τ(Marusic et al. [24]). The different regions/layers in a turbulent channel as a function of reynolds number is given in figure A.1.

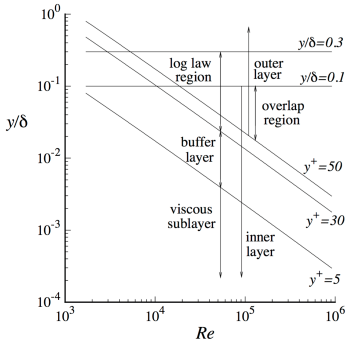


Figure A.1: The regions in a turbulent channel as a function of Reynolds number (Pope [33])

The logarithmic layer or the overlap region consists of both the inner and the outer layer and the flow variables in this region are dependent on both the viscous interactions as well as the characterestics of the outside flow. For high reynolds number, a simplified equation for the mean velocity called the logarithmic law of the wall is used.(Equation A.1)

$$u^+ = \frac{1}{\kappa} \ln y^+ + C^+ \tag{A.1}$$

B

Introduction to Artificial Neural Networks

In order to estabilish the validity of neural networks as a promising tool for SGS models, it is important to gain insight on the working of different kinds of neural networks. This chapter aims to provide to the reader, a good introductory description of neural networks which are relevant to this study. In later parts of the section, related studies in which neural networks are used, are also touched upon. Finally, a discussion of the artificial neural network as an SGS model is presented.

B.1. What are ANN's?

Artificial Neural Network or ANN is a complex information processing system, mathematical model or computational model that is based on the working of a biological nervous system. It is structured using large number of interconnected elements that work together to solve problems. ANN computes results from a series of iteration or runs. It is capable of learning from the outcomes of these previous runs, thereby improving in convergence.Artificial neural networks typically learns from successive iterations with starting point as some initial experience. Unlike normal computer computations neural networks does not just follow a set of instructions to solve a problem, rather it looks for all possible solutions then converging to one solutions based on trial and error. Hence neural networks need an example to be trained on and cannot be designed to perform any task.This signifies that the ANN cannot by itself find a way to solve a particular problem. A simple artificial neural networks typically has a input layer , one or more hidden layers and an output layer as shown in figure B.1.

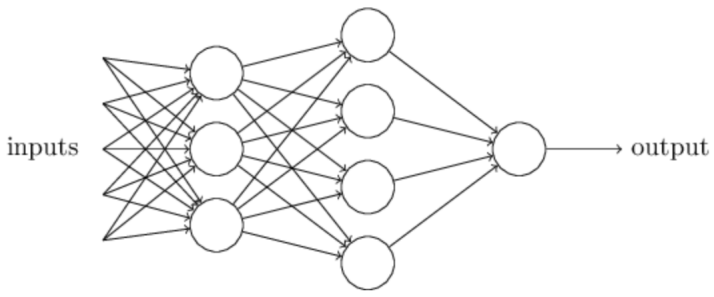


Figure B.1: Schematic representation of a neural network(Nielsen [29])

What is a Neuron?

The ANN as defined is made up of large number of interconnected elements , these elements are called neurons. These neurons as can be seen in image consist of many inputs resembling dendrites

of an actual neuron and one output like the axon. The neurons typically function in two modes called firing mode and training mode. The neuron in training mode learns from the input pattern on when to fire and when not to , It uses this learnt knowledge in the firing mode thereby modifying the input to give a consolidated output. Neurons of complex systems have biases and take weighted inputs, this means that the inputs are summed up disproportionately. These weights and biases are modified for each run, the advantage of doing this is to make sure that new solutions are reached every run. Then based on correlation of this output with the training data a suitable weight and bias is chosen, this weights and biases are characteristic to the problem in hand.

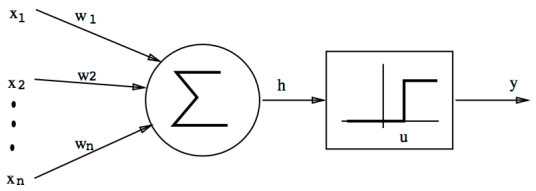


Figure B.2: McCulloch-Pitts Model of a neuron(McCulloch and Pitts [25])

Activation Functions

What goes on behind the hidden layer is largely determine by the so called activation function. The activation function determines how the input is modified and can be as simple as a sum function (linear function) for a simple neural network or a complex non linear function. It is also called as Transfer function. The most commonly used activation functions are linear function, logistic function and hyperbolic tangent.

- Linear Function

$$f(z) = z \tag{B.1}$$

- Logistic Function

$$f(z) = \frac{1}{1+e^{-z}} \tag{B.2}$$

- Hyperbolic Tangent

$$f(z) = tanhz \tag{B.3}$$

It is necessary to note that Beekman [3] used linear functions for the input and output nodes and used the hyperbolic tangent for the hidden nodes.

The Network Build-up

As the network is built with the input layer, output layer and the hidden layer, the connections between the layers are also built. The manner in which the nodes are connected, builds he architecture of the network. The different architectures are explained briefly in section 5.2.1. Each neuron/node is associated with an activation function. Each connection is associated with a weight. These weights are crucial as it determines the contribution of that particular connection to the entire system. As the input moves through, past the input layer (the nodes usually have a linear activation function), it gets multiplied by the weights which are unique to each connected. The input then for the hidden nodes is the weighted input from each connection. Each node is also associated with a bias and this can be thought of as the threshold of that neuron. So the output from the hidden node can be written as shown in equation 5.4

$$output = f(\sum wx + b) \tag{B.4}$$

where f(z) is the relevant activation function, w is the weight of that connection, x is the input and b is the bias of the hidden node.

B.2. Network Layout

The network layout of an artificial neural network differs based on the specific problem. Broadly they can be classified into different types. But what is it that motivates the selection of one typre of network over the other? A brief explanation is given in this section.

B.2.1. Architecture

ANN's can be classified based on the connection pattern of the different nodes. Back in 1996, Jain et al. [20] classified neural networks into two main classes namely,the feedforward networks(where there are no looped connections) and the Recurrent networks (in which the connections were allowed to be feedback as well). This classification is shown in Figure B.3

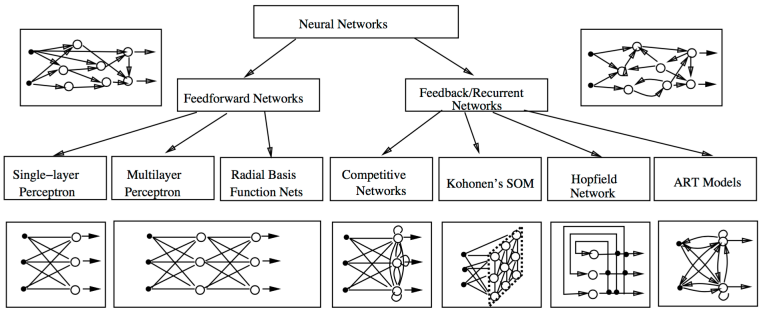


Figure B.3: A taxonomy of network architectures (Jain et al. [20])

Among the feed-forward networks, Multi-Layer Perceptrons gained the most popularity and are known for their approximation capabilities.(Hornik [14]) One of the major contributing factors to the success of MLP's was that the backpropagation training method (explained in section 5.3.1) was effective on this architecture. The MLP's have connections only with succeeding layers which made it easier for training algorithms to minimize the error. A very similar architecture is employed in the Radial Basis function(RBF)networks with the difference only being the type of activation function used in the network. In recent times, Fully connected cascade (FCC) has become a well known architecture for neural networks. FCC requires lesser number of neurons to produce the same results as an MLP, due to its highly interconnected design (as shown in Figure B.4). The different architectures were studied by Hunter et al. [18] and the FCC architecture was shown to be the one which worked most effectively. However the FCC architecture is difficult to implement because of the complex connections. It is impossible to use backpropagation and other training methods are needed, which could be computationally cheap.
Durieux [10] initially used both the MLP as well as the FCC architecture for the initial ANN-based SGS model and his results showed that the MLP architecture had a better performance. Beekman [3] also worked on an MLP architecture and got good results. In this thesis, the MLP architecture is thus chosen.

B.2.2. Network Size

Once the architecture of the neural network is decided, it is necessary to determine the required size of the network for this problem. Increasing the network size will enable increase in mapping capa-

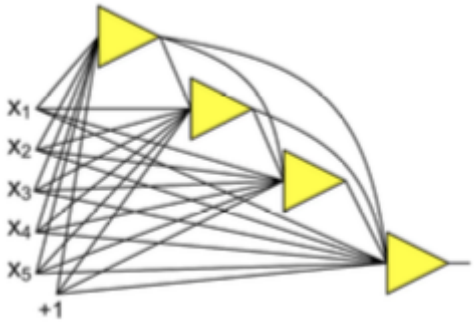


Figure B.4: The FCC topology (Hunter et al. [18])

bilities and increasing accuracy, upto a certain extent. Increasing the size beyond measure however, will lead to reduction in network generalisation capability, as well as an increase in computational cost. It is therefore, necessary to find the optimal number of neurons required for a particular problem. Durieux [10] used just one hidden layer in his computations. The choice was motivated by the study performed by Hornik [14] where he showed that one hidden layer was enough to capture non linear behaviour. Vollant and Corre [42] and Sarghini et al. [36] used two hidden layers for their ANN-based SGS models. Beekman [3] initially performed his study with no hidden layer and later expanded it to one hidden layer with 5 neurons.
Durieux [10] varied the number of neurons in the hidden layer from 1 to 6. He found that the network with 5 hidden neurons showed the most promising results. Vollant and Corre [42] and Sarghini et al. [36] used more number of hidden neurons (18 and 30 respectively)owing to the increased number of variables present in the 3D NSE.

B.3. Training

Through the previous sections it is clear that the architecture used for this study is the Multi-Layer Perceptron(MLP). Training methods often employed on this architecture is discussed in this section. When a neural network is used as a SGS model, there is a need for slight changes. These changes are explained in the subsequent section.

B.3.1. Training Methods

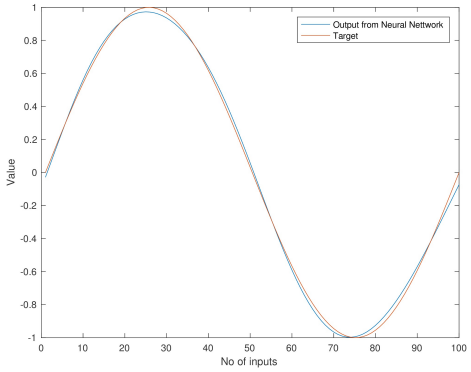
One of the most popular means by which a neural network is trained is backpropagation. This method was first introduced by Rumelhart et al. [35] in 1986, where it is used along with an optimization technique,that is the gradient descent. This framework tends to be an efficient training method when used in neural networks for pattern recognition, classification problems and other problems that require the use of a large data set. (Nielsen [29]). A major disadvantage of basic backpropagation algorithm is that it has a slow learning convergence velocity. Lahmiri [21]. Additionally the performace depends on the learning rate and the complexity of the problem.

Many faster algorithms were then implemented successfully to train a neural network. One of them is the conjugate gradient method. Although the algorithm works similar to that of the gradient descent, it is effectively faster. The reason is that, it performs a search along conjugate directions, and if the direction fails to be a descent direction, the algorithm is restarted using the negative gradient direction to ensure convergence.(Liviers and Pintelas [22]). However it needs the calculation of second derivatives. A completely different approach is done in the Quasi-Newton method where an approximate Hessian matrix is updated after every iteration. The Levenburg-Marquardt algorithm(LMA) is said to combine the benefits of the gradient descent and the best features of the

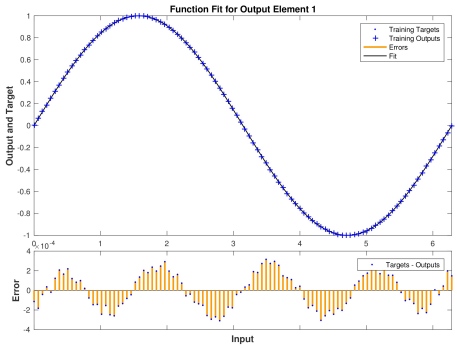
Gauss-Newton method (Lahmiri [21]) . It behaves like a gradient descent method when the solution is far from the correct one. But as the solution approaches the correct one, it becomes a gauss-newton method. LMA is an iterative method and uses a Jacobian instead of a Hessian matrix. It is widely adoped as a minimization technique in many other disciplines as well.Taylor et al. [41].The Levenburg-Marquardt algorithm is considered to be superior when one considers small networks and moderate data sets However, it is not a chosen method for bigger networks and data sets because then the storage and inversion of the Jacobian matrix causes a lot of computational burden.(Beekman [3]).

For the purpose of the current study, the Levenburg-Marquadt Algorithm is chosen as the training method. There are two reasons for this. The network needed for this study would be of small size, and LMA has been proven to have good performance in such cases. The second reason is that, the previous work done by both Durieux [10] and Beekman [3] used this training algorithm and their results is one of the supreme motivations to further indulge into this topic.

Curve-fitting

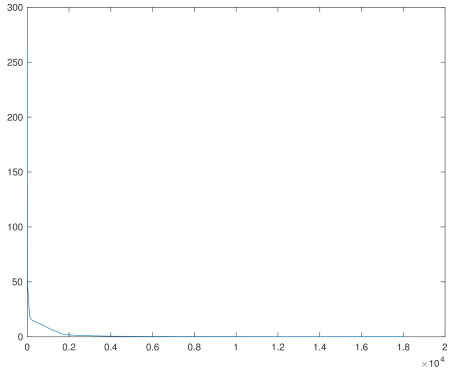


(a) Sine fit

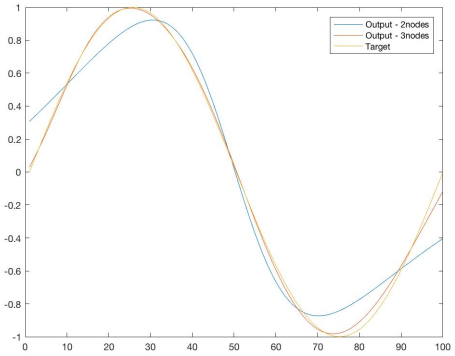


(b) Sine fit using MATLAB toolbox

Figure B.5: The Sine fit using neural networks



(a) Error history plot



(b) Comparison between different number of nodes

Figure B.6: Error history and Comparison

One of the many applications of Neural networks is function approximation.The software MATLAB contains an inbuilt tool with a GUI interface which enables you to create and train a neural network. A simple problem where the neural network is created to fit a sin(x) function is shown in figure 4.5b.

The artificial neural network is fed with increasing values of x as inputs and the corresponding sin(x) is required to be the output. While the MATLAB software facilitates the rapid prototyping and testing of neural architectures,use of a compiled programming language such as C/C++ would provide better run-time performance. Nevertheless, it serves as a platform to learn the basics of neural networks. In MATLAB, a code was written without involving the toolbox to completely understand the working of a neural network. The training method was the gradient descent backpropagation and it showed good results(as shown in figure 5.5a).
The figure 5.5a was obtained when the network consisted of three hidden nodes with one hidden layer. The SGD algorithm was used for training. A low learning rate was required to obtain good results. The number of epochs were made to be quite high. The error graph is shown in figure 5.6a, and the error is reducing with the number of epochs as expected. This shows that the network is learning. The error is the difference between the output and the target value. In figure 5.5b the fit is compared for two nodes and three nodes as per the authors code. The fit is shown to be better for 3 nodes for the same learning rate and number of epochs.

Bibliography

[1] JS Baggett. On the feasibility of merging les with rans for the near-wall region of attached turbulent flows, annual research briefs 1998. *Center for Turbulence Research, Stanford University, Palo Alto, California*, page 267, 1998.

[2] Elias Balaras, Carlo Benocci, and Ugo Piomelli. Two-layer approximate boundary conditions for large-eddy simulations. *AIAA Journal*, 34(6):1111–1119, 1996. ISSN 0001-1452 1533-385X. doi: 10.2514/3.13200.

[3] Matthijs Beekman. Using artificial neural networks as unresolved-scale models for the burgers equation. Report, Delft University of Technology, 2016.

[4] D. H. Chambers, R. J. Adrian, P. Moin, D. S. Stewart, and H. J. Sung. Karhunen–loéve expansion of burgers' model of turbulence. *Physics of Fluids*, 31(9):2573–2582, sep 1988. doi: 10.1063/1.866535. URL https://doi.org/10.1063/1.866535.

[5] D.H. Chambers. *Statistical representations of coherent structures in turbulent flow fields*. PhD thesis, University of Illinois, 1987.

[6] Alexei Chekhlov and Victor Yakhot. Kolmogorov turbulence in a random-force-driven burgers equation. *Physical Review E*, 51(4):R2739–R2742, apr 1995. doi: 10.1103/physreve.51.r2739. URL https://doi.org/10.1103/physreve.51.r2739.

[7] L. Chen, W. N. Edeling, and S. J. Hulshoff. Pod enriched boundary models and their optimal stabilisation. *International Journal for Numerical Methods in Fluids*, 77(2):92–107, 2015. ISSN 1097-0363. doi: 10.1002/fld.3977. URL http://dx.doi.org/10.1002/fld.3977.

[8] Ramon Codina. Stabilized finite element approximation of transient incompressible flows using orthogonal subscales. *Computer Methods in Applied Mechanics and Engineering*, 191(39): 4295 – 4321, 2002. ISSN 0045-7825. doi: http://dx.doi.org/10.1016/S0045-7825(02)00337-7. URL http://www.sciencedirect.com/science/article/pii/S0045782502003377.

[9] Kelly Cohen, Stefan Siegel, Jürgen Seidel, Selin Aradag, and Thomas McLaughlin. Nonlinear estimation of transient flow field low dimensional states using artificial neural nets. *Expert Systems with Applications*, 39(1):1264–1272, jan 2012. doi: 10.1016/j.eswa.2011.07.135. URL https://doi.org/10.1016/j.eswa.2011.07.135.

[10] Thijs Durieux. Exploring the use of artificial neural network based subgrid scale models in a variational multiscale formulation. Report, Delft University of Technology, 2015.

[11] Dr. Richard J. Duro and J. Santos. Modelling temporal series through synaptic delay-based neural networks. *Neural Computing & Applications*, 11(3):224–237, Jun 2003. ISSN 1433-3058. doi: 10.1007/s00521-003-0359-y. URL https://doi.org/10.1007/s00521-003-0359-y.

[12] V. Gravemeier, S. Lenz, and W. A. Wall. Variational multiscale methods for incompressible flows. *International Journal of Computing Science and Mathematics*, 1(2/3/4):444, 2007. ISSN 1752-5055 1752-5063. doi: 10.1504/ijcsm.2007.016545.

[13] Oriol Guasch and Ramon Codina. Statistical behavior of the orthogonal subgrid scale stabilization terms in the finite element large eddy simulation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 261:154 – 166, 2013. ISSN 0045-7825. doi: http://dx.doi.org/10.1016/j.cma.2013.04.006. URL http://www.sciencedirect.com/science/article/pii/S0045782513001047.

[14] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Netw.*, 4 (2):251–257, March 1991. ISSN 0893-6080. doi: 10.1016/0893-6080(91)90009-T. URL http://dx.doi.org/10.1016/0893-6080(91)90009-T.

[15] T. J. R. Hughes, G. R. Feijóo, L. Mazzei, and J. B. Quincy. The variational multiscale method—a paradigm for computational mechanics. *Computer methods in applied mechanics and engineering*, 166(1-2):3–24, 1998.

[16] Thomas J. R. Hughes, Assad A. Oberai, and Luca Mazzei. Large eddy simulation of turbulent channel flows by the variational multiscale method. *Physics of Fluids*, 13(6):1784–1799, 2001. ISSN 1070-6631 1089-7666. doi: 10.1063/1.1367868.

[17] S. J. Hulshoff, E. A. Munts, and J. Labrujere. Wall-stress boundary conditions for variational-multiscale les. *International Journal for Numerical Methods in Fluids*, 66(11):1341–1353, 2011. ISSN 02712091. doi: 10.1002/fld.2311.

[18] D. Hunter, Hao Yu, M. S. Pukish, J. Kolbusz, and B. Wilamowski. Selection of proper neural network sizes and architectures a comparative study. 2012. doi: 10.1109/TII.2012.2187914. URL http://citeweb.info/20120031308.

[19] N. Hutchins and Ivan Marusic. Evidence of very long meandering features in the logarithmic region of turbulent boundary layers. *Journal of Fluid Mechanics*, 579:1, 2007. ISSN 0022-1120 1469-7645. doi: 10.1017/s0022112006003946.

[20] Anil K. Jain, Jianchang Mao, and K. Mohiuddin. Artificial neural networks: A tutorial. *IEEE Computer*, 29:31–44, 1996.

[21] Salim Lahmiri. A comparative study of backpropagation algorithms in financial prediction. *International Journal of Computer Science, Engineering and Applications (IJCSEA)*, 1(4):15–21, 2011.

[22] I.E. Liviers and P Pintelas. A survey on algorithms for training artificial neural networks. 2008.

[23] M. I. A. Lourakis. *A brief description of the Levenberg-Marquardt algorithm implemented by levmar*. Foundation for Research and Technology - Hellas, Vassilika Vouton, P.O. Box 1385, GR 711 10 Heraklion, Crete, GREECE, 2005.

[24] I. Marusic, B. J. McKeon, P. A. Monkewitz, H. M. Nagib, A. J. Smits, and K. R. Sreenivasan. Wall-bounded turbulent flows at high reynolds numbers: Recent advances and key issues. *Physics of Fluids*, 22(6):065103, 2010. ISSN 1070-6631 1089-7666. doi: 10.1063/1.3453711.

[25] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943. ISSN 1522-9602. doi: 10.1007/BF02478259. URL http://dx.doi.org/10.1007/BF02478259.

[26] Charles Meneveau and Joseph Katz. Scale-invariance and turbulence models for large-eddy simulation. *Annual Review of Fluid Mechanics*, 32(1):1–32, 2000. ISSN 0066-4189 1545-4479. doi: 10.1146/annurev.fluid.32.1.1.

[27] M. M. Metzger and J. C. Klewicki. A comparative study of near-wall turbulence in high and low reynolds number boundary layers. *Physics of Fluids*, 13(3):692–701, 2001. ISSN 1070-6631 1089-7666. doi: 10.1063/1.1344894.

[28] Luis Carlos Navarro. Design of residual based unresolved scale models using time averaged solution data. Report, Delft University of Technology, 2015.

[29] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination press edition, 2015.

[30] N. V. Nikitin, F. Nicoud, B. Wasistho, K. D. Squires, and P. R. Spalart. An approach to wall modeling in large-eddy simulations. *Physics of Fluids*, 12(7):1629–1632, 2000. ISSN 1070-6631 1089-7666. doi: 10.1063/1.870414.

[31] F. López Peña, R.J. Duro, and M. Sánchez Simón. Detecting coherent structures in a turbulent wake by using delay based networks. *Computer Standards & Interfaces*, 24(2):171–184, jun 2002. doi: 10.1016/s0920-5489(02)00011-9. URL https://doi.org/10.1016/s0920-5489(02)00011-9.

[32] Ugo Piomelli. Wall-layer models for large-eddy simulations. *Progress in Aerospace Sciences*, 44 (6):437–446, 2008. ISSN 03760421. doi: 10.1016/j.paerosci.2008.06.001.

[33] Stephen N. Pope. *Turbulent Flows*. 11th print edition, 2013.

[34] Aishwar Ravichandran. Investigation of quasi-static, and dynamic orthogonal subgrid scale approximations for variational multiscale method. Report, Delft University of Technology, 2017.

[35] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing, explorations in the microstructure of cognition, volume 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986. ISBN 0-262-68053-X. URL http://dl.acm.org/citation.cfm?id=104279.104293.

[36] F. Sarghini, G. de Felice, and S. Santini. Neural networks based subgrid scale modeling in large eddy simulations. *Computers and Fluids*, 32(1):97–108, 2003. ISSN 00457930. doi: 10.1016/s0045-7930(01)00098-6.

[37] Scovazzi. *Multiscale methods in Science and Engineering*. PhD thesis, Stanford University, 2004.

[38] Farzin Shakib and Thomas J R Hughes. A new finite element formulation for computational fluid dynamics: Ix. fourier analysis of space-time galerkin/least-squares algorithms. *Computer Methods in Applied Mechanics and Engineering*, 87(1):35–58, 1991. ISSN 0045-7825. doi: 10.1016/0045-7825(91)90145-V.

[39] Mikhail L. Shur, Philippe R. Spalart, Mikhail Kh Strelets, and Andrey K. Travin. A hybrid rans-les approach with delayed-des and wall-modelled les capabilities. *International Journal of Heat and Fluid Flow*, 29(6):1638–1649, 2008. ISSN 0142727X. doi: 10.1016/j.ijheatfluidflow.2008.07.001.

[40] S.J.Hulshoff. Cfd 2 : Computation and modelling of turbulence. 2015.

[41] Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, and Tom Goldstein. Training neural networks without gradients, a scalable admm approach. *CoRR*, abs/1605.02026, 2016. URL http://arxiv.org/abs/1605.02026.

[42] Geraci Vollant, G. Balarac and Corre. Optimal estimator and artificial neural network as efficient tools for the subgrid scale scalar flux modeling. pages 1–10, 2014. URL https://hal.archives-ouvertes.fr/hal-01070983.