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

Flow Predictions with Deep Neural Networks

Diploma Thesis

Ioannis Baklagis

Advisor: Kyriakos C. Giannakoglou, Professor NTUA

Athens, 2021

Acknowledgments

I would first like to thank my supervisor, professor Kyriakos C. Giannakoglou. I was honored to be given the opportunity to be a member of the PCOpt/NTUA team, as a research assistant, for almost 3 years. His figure as a scientist, but especially as a teacher, inspired me all the way throughout my studies. I consider myself fortunate to have been working under his supervision in the topics of Computational Fluid Dynamics and Neural Networks. His problem solving experience and his scientific approach in challenging topics altered my engineering insight.

Secondly, I am wholeheartedly thankful for the support that I received from the members of PCOpt/NTUA team. Despite their busy schedule, they were always available to share their knowledge and eager to assist me. I would like to express my sincere gratitude to Dr. Dimitrios Kapsoulis, who introduced me and inspired me in the topic of Neural Networks. I am thankful to Dr. Varvara Asouti for her support and to Marina Kontou for the excellent cooperation during the joined publications.

Thirdly, I would like to thank my parents for always being there, believing in me and supporting me with their love. I am thankful to my friends for accompanying me along the journey creating glorious memories and unforgettable experiences. I would also like to thank A. for her support.

In loving memory of my grandfather, Stavros, my inspiration for becoming a mechanical engineer.

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

Flow Predictions with Deep Neural Networks

Diploma Thesis

Ioannis Baklagis

Advisor: Kyriakos C. Giannakoglou, Professor NTUA

Athens, 2021

Abstract

Artificial Intelligence (AI) has developed rapidly in recent years. It has penetrated deeply into the daily life of the average human, as well as the industry and academia. Deep Neural Networks (DNNs), which are part of the field of AI, are interconnected neural networks that have the ability to make predictions by presenting them with the corresponding input. Inhere, two types of DNNs are utilized, the Long Short-Termo Memory (LSTM) networks and the λ-DNN.

Firstly, a quasi-1D flow in a human artery is modelled and the 1DAS software is created, which solves the blood flow numerically. The initial artery shape and the wall thickness longitudinal distribution vary, while the blood inflow is time-varied (pulsatile heart action). The software generates time-varying longitudinal distributions of velocity, cross-sectional area and, pressure, creating a training dataset for the network. The LSTM network is, then, trained to predict the following velocity distribution by presenting it with the preceding ones of the same quantity. Its architecture and input are, later, optimized through a population-based algorithm and a statistical method, respectively. Two benchmark cases (simple periodic function and heat conduction equation) demonstrate the capabilities of the LSTM network, as well.

Later in this diploma thesis, the λ-DNN is utilized for predicting aerodynamic flows and temperature distributions. It is a multi-branch architecture, with its input consisting of nodal coordinates and case-related data. Firstly, the network is trained to predict the pressure field around an isolated airfoil. Secondly, the same network is trained to predict the pressure distribution on the surface of a Francis runner. Lastly, the network is implemented in a multi-disciplinary problem, namely a Conjugate Heat Transfer (CHT) one. The λ-DNN is trained to replicate the solver of the heat

conduction equation (one discipline) and predicts the temperature distribution on the contour of an internally cooled blade. Its capabilities are evaluated by a short presentation of an optimization that utilizes its predictions.

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

Πρόβλεψη Ροών με Βαθιά Νευρωνικά Δίκτυα

Διπλωματική εργασία

Ιωάννης Μπακλαγής

Επιβλέπων: Κυριάκος Χ. Γιαννάκογλου, Καθηγητής ΕΜΠ

Αθήνα, 2021

Η Τεχνητή Νοημοσύνη (ΤΝ) έχει αναπτυχθεί ραγδαία τα τελευταία χρόνια και έχει εισχωρήσει βαθύτατα στην καθημερινή ζωή του μέσου ανθρώπου, καθώς και στη βιομηχανία και τον ακαδημαϊκό χώρο. Τα Βαθιά Νευρωνικά Δίκτυα (ΒΝΔ), που ανήκουν στο πεδίο της ΤΝ, είναι διασυνδεδεμένα δίκτυα νευρώνων που έχουν τη δυνατότητα να κάνουν προβλέψεις τροφοδοτούμενα με την αντίστοιχη είσοδο. Στη διπλωματική αυτή εργασία χρησιμοποιούνται δύο είδη δικτύων, το Long Short-Term Memory (LSTM) δίκτυο και το λ-DNN.

Αρχικά, μοντελοποιείται η ψευδο-μονοδιάστατη ροή αίματος σε μοντέλο ανθρώπινης αρτηρίας και δημιουργείται το λογισμικό 1DAS, το οποίο επιλύει αριθμητικά τη ροή. Το αρχικό (απλοποιημένο) σχήμα της αρτηρίας και η διαμήκης κατανομή πάχους του τοιχώματος μεταβάλλονται χωρικά, ενώ η εισροή αίματος μεταβάλλεται χρονικά (παλμική καρδιακή δράση). Το λογισμικό δημιουργεί χωρικές κατανομές ταχύτητας, διατομής και πίεσης που μεταβάλλονται στο χρόνο, δημιουργώντας ένα σύνολο δεδομένων εκπαίδευσης για το δίκτυο. Το LSTM δίκτυο εκπαιδεύεται για να προβλέψει την κατανομή ταχύτητας σε κάθε επόμενη χρονική στιγμή τροφοδοτώντας το με τις προηγούμενες. Η αρχιτεκτονική και οι είσοδοι του δικτύου βελτιστοποιούνται μέσω ενός αλγόριθμου ελαχιστοποίησης που χειρίζεται πληθυσμούς υποψήφιων λύσεων και μιας στατιστικής μεθόδου, αντίστοιχα. Δύο εφαρμογές (απλή περιοδική συνάρτηση και εξίσωση αγωγής θερμότητας) καταδεικνύουν επίσης τις δυνατότητες του δικτύου LSTM.

Στην επόμενη φάση της εργασίας, το λ-DNN χρησιμοποιείται για την πρόβλεψη αεροδυναμικών ροών και κατανομών θερμοκρασίας. Η αρχιτεκτονική του δικτύου είναι μία αρχιτεκτονική πολλαπλών κλάδων, με την είσοδό του να αποτελείται από κομβικές συντεταγμένες και δεδομένα που εξαρτώνται από την εκάστοτε εφαρμογή. Αρχικά, το δίκτυο εκπαιδεύεται να προβλέπει το πεδίο πίεσης γύρω από μία αεροτομή. Έπειτα, το ίδιο δίκτυο εκπαιδεύεται να προβλέπει την κατανομή πίεσης στην επιφάνεια ενός δρομέα ενός υδροστροβίλου τύπου Francis. Τέλος, το δίκτυο χρησιμοποιείται σε ένα

πολυπεδιακό πρόβλημα, και, πιο συγκεκριμένα, το πρόβλημα Συζευγμένης Μεταφοράς Θερμότητας (ΣΜΘ). Το λ-DNN εκπαιδεύεται να αντικαταστήσει τον επιλύτη της εξίσωσης αγωγιμότητας θερμότητας (ένα πεδίο του πολυπεδιακού προβλήματος) και να προβλέπει την κατανομή της θερμοκρασίας στο περίγραμμα ενός εσωτερικά ψυχόμενου πτερυγίου. Οι δυνατότητές του αξιολογούνται με μια σύντομη παρουσίαση μιας βελτιστοποίησης που χρησιμοποιεί τις προβλέψεις του.

Nomenclature

NTUA	National Technical University of Athens
PCopt	Parallel CFD & Optimization unit
AI	Artificial Intelligence
ML	Machine Learning
ANN	Artificial Neural Network
DNN	Deep Neural Network
CFD	Computational Fluid Dynamics
CNN	Convolutional Neural Network
GAN	Generative Adversarial Network
LSTM	Long Short-Term Memory
RBF	Radial Basis Function
MAEA	Metamodel Assisted Evolutionary Algorithm
MAE	Mean Absolute Error
MAPE	Mean Percentage Absolute Error
RNN	Recurrent Neural Network
A	Cross-Sectional Area
u	Velocity
p	Pressure

A_0	Initial Cross-Section Area of the Artery
$\dot{Q}_{inlet}$	Blood Inflow of the Artery
WK	Windkessel Model
CP	Control Point
R_{art}	Artery Radius
T_h	Time Period of Heart Rate
ρ_b	Density of Blood
P_{ext}	External Pressure around the Artery Wall
E	Young Modulus of Artery Wall
σ	Poisson Ratio of Artery
h	Thickness of Artery Wall
p_c	Initial pressure of Capacitor (WK)
C_s	Capacitance (WK)
R_1	Resistance R1 (WK)
R_2	Resistance R2 (WK)
L_{art}	Length of the Artery
1DAS	1-D Arterial Solver
T	Temperature of Plate
L_p	Width of Plate
T_{in}	Internal Temperature
T_{out}	External Temperature

h_{in}	Convective Heat Transfer Coefficient - Internal
h_{out}	Convective Heat Transfer Coefficient - External
k	Coefficient of Thermal Conductivity of Plate
C_{pl}	Thermal Capacity of Plate
ρ	Density of plate
n_{steps}	Number of Time Instants

Contents

Chapter 1

Artificial Intelligence

1.1 Artificial Intelligence and Machine Learning

In recent decades, there has been a rapid development in the field of computer systems. Computers are able to perform many calculations very fast and on a large scale. They have dominated human life and often tend to replace jobs/tasks traditionally done by humans. Today's youth can not imagine their lives without the automatic translation (i.e. Google Translate), [1], or personalized recommendation in video streaming applications. Smartphones are becoming smarter day by day. Smart watches are capable of notifying their user for irregular cardiac rhythm, [2]. Text-to-speech and speech-to-text features are of the utmost importance to some people. Cars are revolutionized by computer science with the incorporation of self-driving, [3]. Each one of these examples highlights the domination of the intelligence demonstrated by machines, also known as Artificial Intelligence (AI).

Machines are programmed to simulate human intelligence by acting like them and mimicking their behaviour (cognitive activity). They can learn to recognize complex patterns in big data, make decisions in challenging problems and, execute tasks. A great benchmark for validating the current progress and state of AI are games created for humans (i.e. Chess, Go). The *DeepBlue* computer, designed by *IBM*, was able to beat chess grand master Garry Kasparov at the game in 1997. *Alpha Star* is bot that plays the game *StarCraft II*, [4]. The game is a fast-paced multiplayer real-time strategy game developed and published by *Blizzard Entertainment* and it is consider as one of the hardest and most challenging strategy games. The *Alpha Star* program was created by the British artificial intelligence subsidiary of *Alphabet Inc.*, called *DeepMind*. In January 2019, the program managed to win two professional players, [5], and in October, of the same year, reached the top league, becoming the first AI to advance in this position, [6].

Thus, the applications of AI are endless and can be in many sectors. As industrial and academic demands have increased, their problems have become more complex. It is reasonable, for the industry and academia, to turn to AI to assist them in problem-solving and decision-making, utilizing artificial learning, reasoning, and perception. In terms of engineering, AI helped to improve tasks or even to overcome previously unresolved problems. Consider how Computer Aided Engineering (CAE) is a fundamental tool of mechanical engineering, and it was once just a supplemental software. Correspondingly, AI penetrates the life of the engineer. Many papers are published every day that showcase implementations of AI in various engineering topics, such as CAD, [7], mechanical systems with gears, [8] and, thermal systems, [9].

Machine Learning (ML), fig. 1.1, is an application of AI that has the ability to acquire its own knowledge (learn) by presenting it with raw data, [10]. ML models are capable of learning automatically by extracting patterns from sample data. As a baby learns to distinguish a dog from a cat by presenting examples from each one, so a sophisticated algorithm can recognize objects or faces by presenting it with labelled or not labelled samples. ML can be categorised:

- **Supervised Learning**: The model is presented with labelled samples, with input and the corresponding output, which it is trained to predict.
- **Non-Supervised Learning**: The model is called to map patterns in input data without presenting it with the output (unlabeled data).
- **Reinforcement Learning**: The model is trained to make decisions that either reward it or punish it. Its goal is to maximize the reward by completing challenging tasks, such as navigating a robot in a room.

Deep Learning (DL) is a method of ML that is based on Artificial Neural Networks (ANN), [11]. An ANN is a ML model consisting of units, called nodes or neurons, that is presented with labeled data. It is trained to predict the desired output by minimizing the error between it and its prediction. Deep Neural Networks (DNNs) are ANN with many layers of neurons. ANNs and DNNs are further discussed in Chapter 2.

1.2 Artificial Intelligence in CFD and Optimization

DNNs have been widely exploited in the field of Computational Fluid Dynamics (CFD), [12]. CFD codes are utilized today to solve more computationally expensive problems. Despite the rapid development in GPUs and CPUs, the demand for computational resources has rocketed. The ability of DNNs models to extract pat-

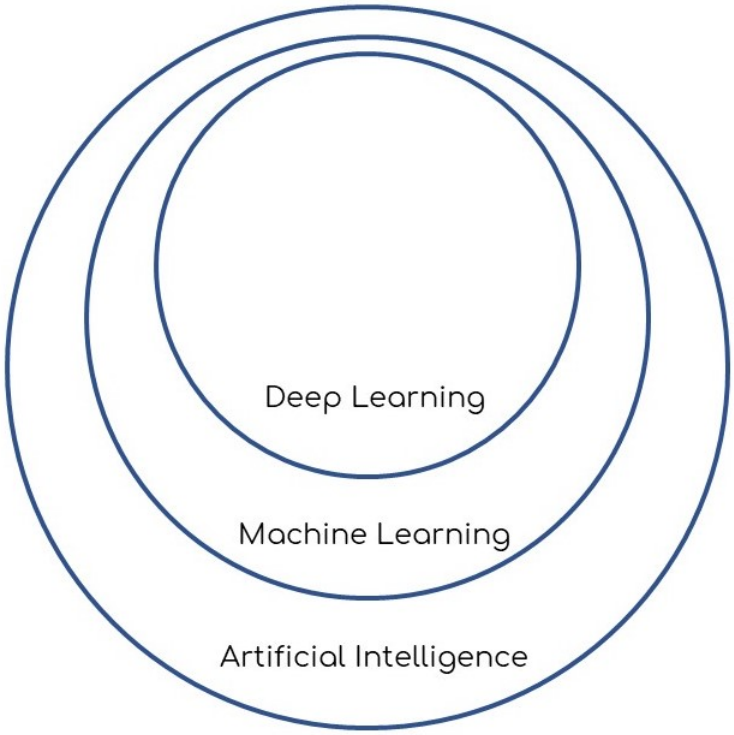


Figure 1.1: *Machine Learning and Deep Learning as subfields of Artificial Intelligence and Machine Learning, respectively.*

terns in raw data and, map the correlation between their input and output, makes them the perfect tool for replicating or assisting these expensive codes. In addition to that, when industries are creating and testing new designs, they produce training data available for training the ML models. For instance, Convolutional Neural Networks (CNNs) were utilized for approximating steady velocity fields, [13]. A symmetrical DNN (normal and transposed convolutional layers, max-pooling layers and, fully connected layers) was used for reconstructing the flow field structure by presenting it with the discrete pressure coefficient distribution on the wall surface of the cascade channel, [14]. In [15], CNNs were used to predict aerodynamics flow fields. A Generative Adversarial Network (GAN) combined with CNNs, called ffs-GAN, predicted transonic flow field profiles of parameterized supercritical airfoils, [16].

Inhere, the DNNs are implemented in various cases for predicting flows. Two different types of DNNs are utilized, the LSTM networks and the λ-DNN. The training patters are generated by CFD software, created either by the author or by the members of the PCOpt/NTUA team. The DNNs are used for reconstructing biological or aerodynamic flows and, for replicating a discipline, in aero-thermal analysis.

In engineering optimization procedures, in order to reach the optimal design, it is required to evaluate many designs. In computational mechanics, the evaluation of

shapes, to be optimized, claims the run of time-consuming and computationally expensive simulation codes. Especially, when a stochastic optimization (i.e. Evolutionary Algorithm) is in use, the number of shapes is highly increased. Thus, DNNs are perfect candidates for replicating these expensive CFD codes and, they are capable of acting as surrogate models during shape optimization. These DNNs are often referred as metamodels. In[17] and [18], Gaussian processes and radial basis function networks (RBF) were used in Metamodel Assisted Evolutionary Algorithms (MAEAs) as metamodels. [19] proposed a DNN that was utilized in MAEA optimization of an isolated wing, maximizing its lift. Inhere, a MAEA optimization is presented, shortly, by exploiting the λ-DNN.

1.3 Thesis outline

Following the introduction, the chapters composing this thesis are presented:

- **Chapter 2**: A gentle introduction is made to how Deep Neural Network work. The model of the neuron, the building blocks of the neural networks, and the training elements are explained. Recurrent Neural Networks and Long Shot-Term Memory networks are presented, as well.
- **Chapter 3**: The mathematical foundation of the quasi-1D arterial blood flows is presented. The system of equations is numerically solved and the 1DAS software is created and showcased.
- **Chapter 4**: Two benchmark cases are introduced for evaluating the LSTM network capabilities. In the first case, the network predicts a periodic function. In the second case, the network predicts temperature distributions of a plate.
- **Chapter 5**: Long Shot-Term Memory neural networks are exploited for predicting time-varying flows in quasi-1D arteries. In each case, the distributions of cross-sectional area and wall thickness vary in order to evaluate the performance and the capabilities of the network.
- **Chapter 6**: Another type of network, the λ-DNN, is used for predicting scalar fields in three cases. In the first two cases, the network is trained to predict the pressure distribution around an isolated airfoil and on the surface of a Francis turbine runner. In the third case, network is trained to replicate the heat conduction equation solver in a multi-disciplinary problem, known as Conjugate Heat Transfer. The network is trained to predict the temperature distribution along the contour of an internally cooled turbine blade.

Chapter 2

Deep Neural Networks

2.1 Artificial Neural Networks

Artificial Neural Networks are computing systems that mimic the biological neural networks and how they function. They fall into the field of Artificial Intelligence and in particular that of Machine Learning. ANNs have been widely used in a variety of applications, such as speech recognition, [21], computer vision, [20], medical diagnostics, [22], automatic translation, [1], and even in activities traditionally considered to be exclusively human, such as painting, [23].

Modern computers attempt to mimic the brain in its complexity and non-linearity. The human brain has the ability to organize its components (neurons), fig. 2.1, so that it can perform calculations fast, make decisions, and develop its own behaviour (experience), [24]. This experience allows the human to adapt to its environment, keeping him alive. So, ANNs, made up of artificial neurons, attempt to mimic the way the brain performs a specific task.

An ANN is a network of interconnected neurons, whose goal is to solve a computational problem. ANNs use an interface of simple computational nodes, referred to as neurons. The network has a tendency to store experience/knowledge and use it later for decision-making and information-processing. This experience is acquired through a learning process, called training, by presenting it with the input values (or signal) and the corresponding output ones (desired target of the ANN). The connection weights between neurons, namely the synaptic weights, are used to store this information. Each neuron is capable of receiving an input, processing it, and producing an output that transmits to other neurons connected to it. Note that the synaptic weights adjust as training progresses, while the weight increases or decreases the output strength in a synaptic connection.

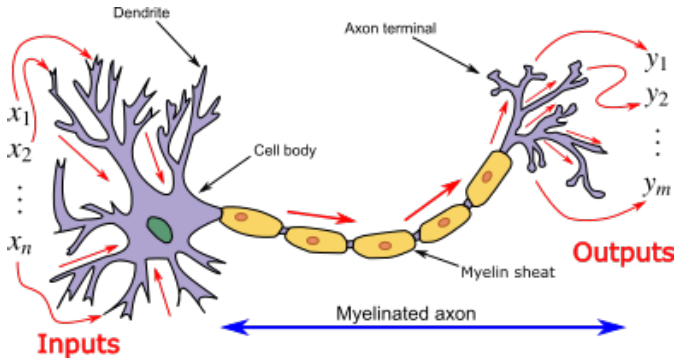


Figure 2.1: *Neuron and myelinated axon. The signal flows from dendrites to axon terminals. From: Egm4313.s12 at English Wikipedia, CC BY-SA 3.0* `https://creativecommons.org/licenses/by-sa/3.0`*, via Wikimedia Commons*

2.2 Neuron Model and Neural Networks

The building blocks of the ANNs are the neurons. A neuron is an information processing unit. It is modelled, fig. 2.2, by a set of 4 elements, [24],

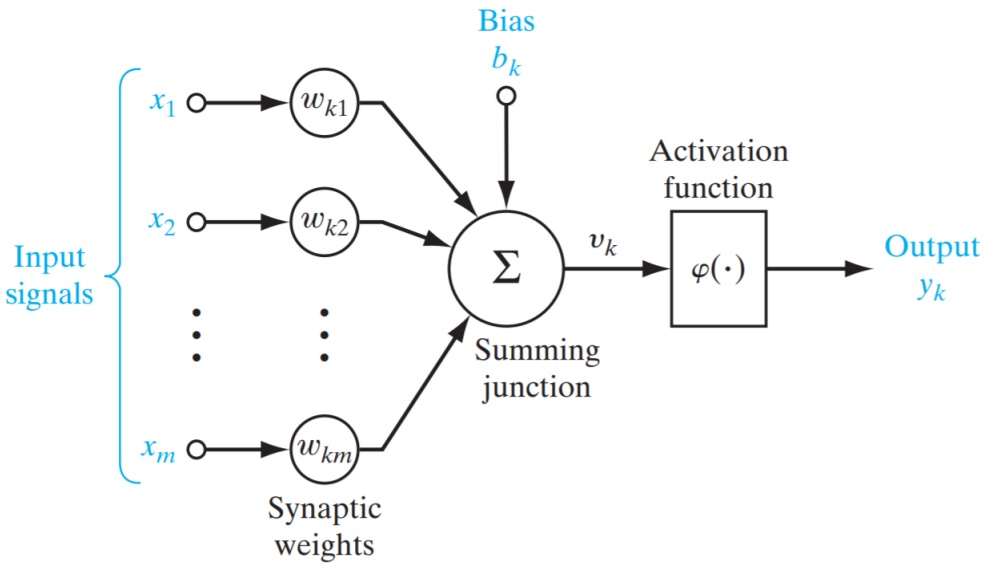


Figure 2.2: *Model of neuron k. From [24]*

- A set of *synaptic weights* (w). The input x_j that is being fed to neuron k is multiplied by the synaptic weight w_{kj}.
- A *summing junction* (u_k) that sums the weighted inputs.
- An *activation function* ($\phi(\cdot)$) that limits the output, increasing or decreasing the neural connection strength.

- A *bias* (bk) of the neuron k.

or in terms of mathematical relations

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.1}$$

and

$$y_k = \phi(u_k + b_k) \tag{2.2}$$

A neural network, fig. 2.3, is a directed graph of interconnected neurons/nodes with synaptic weights and activation functions, [24]. Neurons are organized into layers, where neurons of each layer connect only (except in special cases) to all neurons of the immediately preceding and immediately following layers. The layer that receives input data is the input layer, while the one that produces the network output/prediction is the output layer. Between them, there can be other layers, namely the hidden layers. Their name comes from the fact that they do not come in direct contact with either the input or output. By adding one or more hidden layers, the network can extract higher order information from its input, due to the increase of the synaptic connections. When the network consists of more than 2 hidden layers, then it is called Deep Neural Network (DNN). Finally, a feedforward neural network is a ANN whose connections between nodes do not form a circle (unlike Recurrent Neural Networks) and, the information is directed from the input layer, flowing successively from all hidden layers, to the output one.

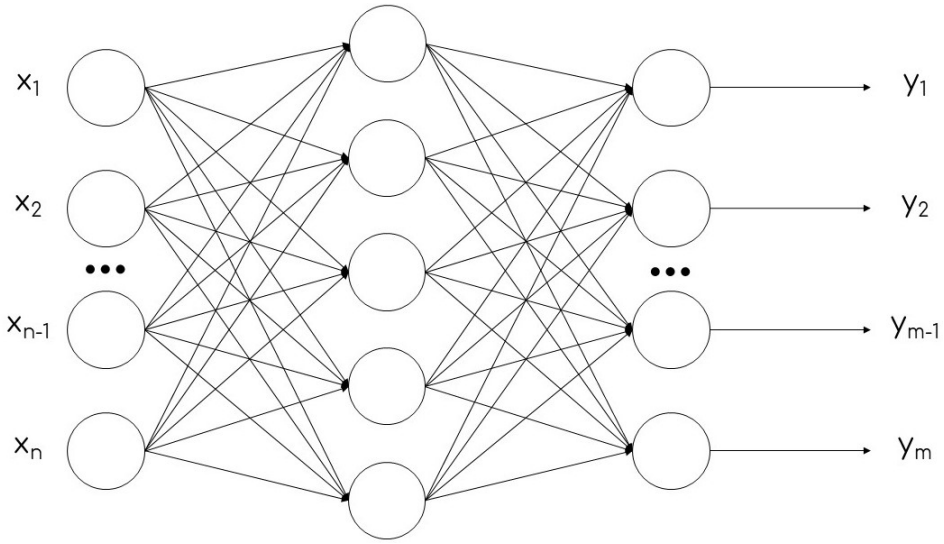


Figure 2.3: *A feedforward neural network. It consists of one hidden layer. It has n inputs (x) and m outputs (y).*

2.3 Training

The training is carried out by presenting the network with input values and the corresponding output ones, which it tries to predict. The learning algorithm performs an optimization by adjusting the synaptic weights of the network smoothly. This is done by minimizing the calculated error between the network prediction and the desired output, with respect to the synaptic weights. Thus, it is required to create a database (data set) with a correspondence of input and output values, which are called patterns/samples. It is divided into two sets, one for training (training data set) and one for network evaluation (validation data set). Usually, the ratio is 8 to 2, i.e. in 10 samples, 2 are used for evaluation and the remaining 8 for training, [10]. In addition, the set used for training is divided into smaller sets of the same number of samples, which are called batches. The number of samples per batch is called batch size and affects the speed as well as the accuracy of the training. Batch size is the number of samples presented to the network before correcting the synaptic weights. Finally, the number of epochs determines how many times the entire data set will be presented to the network (ie only the training data set).

2.4 Architecture

The two main design variables of the architecture are the width and the depth of the network. Width refers to the number of neurons per layer, while depth refers to the number of layers. As the parameter space of the learning algorithm increases (the number of weights), the algorithm can learn more complex patterns and extract more features. At the same time, however, the algorithm is more prone to overfitting and its generalization capability is likely to decrease. Overfitting is when a neural network has learned the training dataset too well. The network is capable of making accurate predictions only in the training dataset and performs poorly on any other set of patterns.

Networks with few layers and many neurons (very wide and shallow networks) are prone to overfitting. Networks with multiple hidden layers (deep networks) are much better at generalizing, since they may extract high-order features in between layers. For instance, if a CNN is called to classify images of human faces, its first layer will be trained to recognize shape edges. The second layer will recognize shapes and the third will recognize a set of shapes composing the nose, etc. In each layer, an extraction of higher-order features/information takes place and, the network can easily map the correlation between the input and the output. Thus, the selection process of the network architecture involves several stages, utilizing the trial and error method. Initially, a network with very few layers is created. Then, more layers are added, with a small number of neurons. At the same time, the number of neurons increases at each layer, until a network with acceptable error is reached

without overfitting.

2.5 Hyperparameters

Hyperparameters are fixed network parameters whose values are set a priori the learning process and are usually determined empirically. Examples of hyperparameters comprise the learning rate, the number of hidden levels, epochs and the batch size. The values of some hyperparameters are dependent on the values of other hyperparameters. For instance, the value of the learning rate may depend on the total number of layers. The hyperparameters determine the network architecture and how it is trained. Their values are selected by the user, while the network parameters (e.g. synaptic weights) are adjusted, automatically, during training.

2.6 Cost Function

The goal of the network training is to minimize the Cost Function. It is related to the Loss Function that accounts for the error between the DNN output and the target for the corresponding input values, for a single training pattern. On the other hand, a Cost Function can contain many Loss Functions (or their average) and it describes all the training data. An example of a Cost Function is Mean Absolute Error

$$MAE = \frac{\sum_{i=1}^{n}(\mid y - y_{tar} \mid)}{n} \tag{2.3}$$

where y is the output of the DNN, y_{tar} is the ideal output of the DNN for the corresponding input and n is the number of predictions. The Loss Function of the MAE is the Absolute Error (AE)

$$AE = \mid y - y_{tar} \mid \tag{2.4}$$

Thus, the MAE Cost Function is the average of the AE Loss Function. In the literature, the terms Loss Function and Cost Function are usually synonymous, if not identical. Note that Mean Absolute Percentage Error (MAPE) may also be used as Cost Function,

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}(\mid \frac{y - y_{tar}}{y_{tar}} \mid) \tag{2.5}$$

2.7 Back Propagation and Optimization Algorithm

Back Propagation, [25], is an algorithm used during the training of NN. It calculates the gradient of the cost function with respect to the synaptic weights. The amount of error is distributed (back propagated) between the neural connections, in proportion to how much they contribute to the computed error. It is called back propagation because it is applied repeatedly in reverse order with the flow of information, starting from the output layer and continuing to the input layer.

The optimization algorithm uses the computed gradient (from the back propagation algorithm) to perform the gradient descent. Note that the derivative is a vector in the opposite direction from the desired minimum. The goal of the algorithm is to minimize the cost function by adjusting the synaptic weights. The Adam optimization algorithm, [26], is a specialized stochastic gradient descent algorithm and is one of the most common optimization algorithms in DNN training. Since it is an efficient algorithm, it is well suited for large problems (in terms of parameters or/and data). It uses the calculated gradient, as well as its statistics and previous values (during training) in order to optimize the weights. The Adam algorithm combines the advantages of the Adaptive Gradient Algorithm (AdaGrad), [27], and Root Mean Square Propagation (RMSProp), [28].

2.8 Learning Rate

The learning rate is a hyperparameter that determines the size (the order) of the corrective update to the network weights, at each iteration during its training, to minimize the error between its output and the desired values. It is a weight (in form of a factor) applied to the correction of the synaptic weights during the network training. A higher learning rate decreases the training time, but with decreased final accuracy. In contrast, a lower rate increases the training duration, although with the possibility of a greater accuracy.

2.9 Recurrent Neural Networks

A Long Short-Term Memory (LSTM) neural network, [29], which is later exploited for blood flow predictions, is a type of Recurrent Neural Network (RNN) architecture, [25]. RNNs are a type of neural network that process sequential data and can scale to much longer sequences. RNNs connect the outputs of all neurons to the input of all neurons, fig. 2.4. The current input and output is influenced by the previous input and output (previous elements in the sequence). This directional information forms a type of "memory" in weights of the RNN, which are adjusted

during the training.

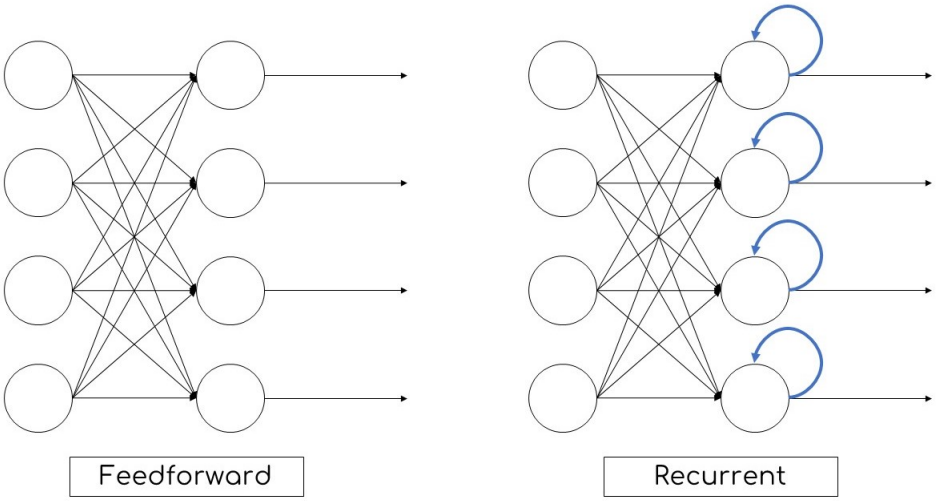


Figure 2.4: *The comparison between the classic Feedforward Neural Network (left) and Recurrent Neural Network (right).*

If the RNN is unfolded, fig. 2.5, it is easier to understand the significance of directing the output to itself. In order to perform a prediction at a time step t in sequential data, the network utilizes the previous outputs. Highlight that the output of previous time steps is dependent on the previous inputs and carries out information through the time. The network creates a form of "memory", which is critical for temporal or ordinal problems, such as speech recognition, [30] and handwriting recognition, [31].

RNNs share the same weights within each layer compared to a typical feedforward network, in which the weight varies in each node. Thus, the Back Propagation is altered to Back Propagation Through Time (BPTT), which is a generalized form of the algorithm, used during the network training. The most common drawback of RNNs are the vanishing or exploding gradient problems. The error, which is back propagated, tends to vanish or explode, preventing the training. When the gradient is decreased, it continues to shrink, and, thus, the network is unable to train. This is known as the vanishing gradient problem. On the other hand, the exploding gradient problem is when the gradient becomes too large and, the model becomes unstable, resulting in weights with infinite values.

2.10 Long Short-Term Memory Networks

LSTM networks were introduced as a solution for the vanishing gradient problem. The presence of long-term dependencies in the input sequence leads to inaccurate predictions from the RNN. Practically, if the prediction of the current state is de-

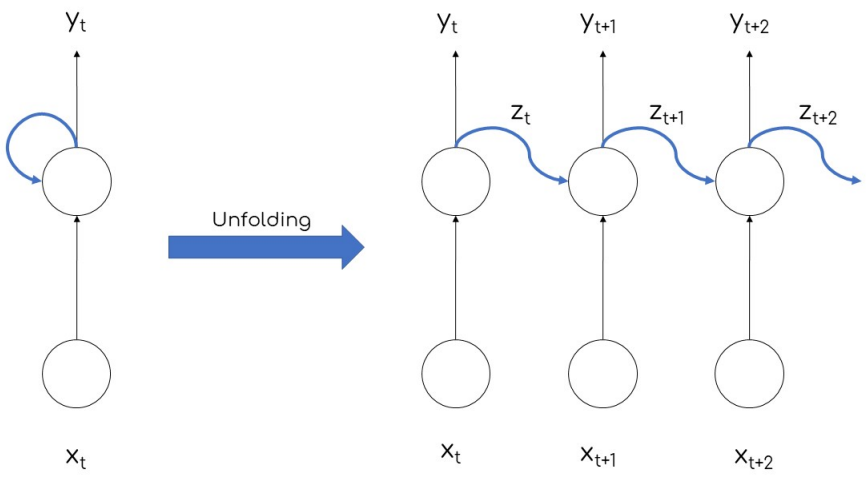


Figure 2.5: *An unfolded RNN. The state at time t influences the future states. The layers are different steps in time of the same RNN.*

pendent on a previous one and this previous state was not in the recent past, the RNN is unable to predict the current state.

The aforementioned problems were addressed and solved, [29], by introducing the LSTM unit. This unit, fig. 2.6, is capable of learning long-term dependencies and comprises a memory cell and three gates, an input gate, an output gate and a forget gate. The cell stores the acquired information through time and the three gates control the flow of information.

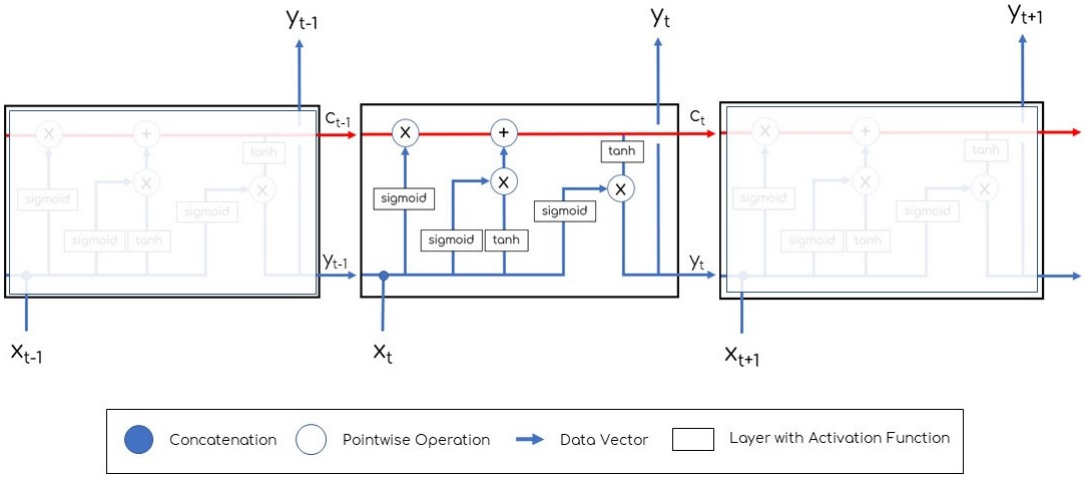


Figure 2.6: *An unfolded LSTM unit. It comprises the cell (red line) and three gates. The basic vectors are the input (x_t), the output (y_t) and the cell state (c_t).*

In fig. 2.6, the top line that runs through the unit is the cell, which carries out

the information along the LSTM unit. The LSTM unit is capable of regulating the information in the cell by adding or removing information, using the gates with pointwise operations. The inputs of the cell consist of its previous cell state (c_{t-1}), its previous prediction (y_{t-1}) and the current input (x_t).

Firstly, the unit decides what information will forget. On the bottom left, the two inputs (x_t and y_{t-1}) are concatenated into a vector that is copied in every branch that is passed to. In the first branch, the resulted vector passes through a sigmoid layer, the forget gate layer. Secondly, the unit decides what information will be "memorized" by the cell. The second branch of the concatenated vector passes through a sigmoid layer, called the input gate layer. This branch, actually, decides what values of the current state cell will be updated. The third branch passes through a tanh layer, which proposes values for the cell state. The second and the third branches are multiplied pointwise, resulting into a proposed vector for updating the cell state. The new, updated, cell state (c_t) results from the multiplication of the old cell state with the first branch and the addition to it of the proposed vector.

The output of the unit (y_t) is influenced by the current state. The concatenated vector of the two inputs (x_t and y_{t-1}) is passed through a sigmoid layer that decides what values of the current state will be utilized for the output. The current cell state passes through a tanh layer and is multiplied by the output of the sigmoid layer. This resulted vector, the output of the LSTM unit, is the combination of the current cell state, the current input and the previous prediction. Note that in every sigmoid and tanh layer a bias is added.

2.11 Implementation

Data pre- and post-processing as well as the DNN implementations are carried out in Python 3.6 utilizing an open source machine learning library, called Tensorflow, [32], all running on CPUs or GPUs, such as K20, V100 and 1050, manufactured by NVIDIA.

Chapter 3

Quasi-1D Artery Blood Flow

A quasi-1D flow problem in an artery with elastic walls is modelled. The artery has varied wall thickness along its length. Its longitudinal initial cross-sectional area distribution is varied, as well. The time-varying blood inflow represents the pulsatile action of the human heart. The generated biological flows are later used as patterns for training the LSTM network.

3.1 System of Governing Equations

As presented in [33], continuity of mass and momentum leads to the following equations in terms of the cross-sectional area (A), the mean velocity (u) over a cross-section and the internal pressure (p) of the artery,

$$\frac{\partial A}{\partial t} + \frac{\partial (Au)}{\partial x} = 0 \tag{3.1}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{1}{\rho}\frac{\partial p}{\partial x} - \frac{1}{\rho}\frac{\partial \tau}{\partial x} = 0 \tag{3.2}$$

where ρ is the constant blood density and τ is the shear stress.

For the shear stress gradient, the expression for a Poiseuille flow,

$$\frac{d\tau}{dx} = -\frac{8\mu\dot{Q}}{\pi R^4} = -\frac{8\pi\mu u}{A} \overset{\mu^* = \frac{8\pi\mu}{\rho}}{=} -\frac{\mu^* \rho u}{A} \tag{3.3}$$

is used where $\dot{Q} = Au$ is the volume flowrate, R is the inner artery radius and μ the

blood viscosity. Note that the expression of eq. (3.3) is not valid for non-Newtonian or turbulent flows.

By incorporating eq. (3.3) into eq. (3.2), the system of two (rather than three) equations (in conservation/vector form), governing the blood flow in an artery with elastic walls, is derived,

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = \mathbf{S} \tag{3.4}$$

where

$$\mathbf{U} = \begin{bmatrix} A \\ u \end{bmatrix}, \ \mathbf{F} = \begin{bmatrix} uA \\ \frac{u^2}{2} + \frac{p}{\rho} \end{bmatrix} and \ \mathbf{S} = \begin{bmatrix} 0 \\ -\frac{8\pi\mu u}{\rho A} \end{bmatrix}$$

Since there are 2 equations and 3 variables (A, u, p), a third equation is necessary to close the system. This third equation describes the vessel wall behaviour due to pressure changes and its difference between the inside of the artery and the surrounding tissue. It introduces the adaptation of the cross-sectional area to the changing internal pressure and deals with the fluid-structure interaction.

Many different models that simulate the relation between the pressure and the cross-sectional area are in use. These are classified as:

- Linear elastic: The area is linearly related to pressure, [34].
- Non-linear elastic: The area is non-linearly related to pressure, [35].
- Collapsible tube: The area is related to pressure with a "tube law" in which the tube can collapse and distend, [36].
- Visco-elastic: Viscoelastic behaviour of the elastic walls is being considered, [37].

In this study, the equation of the non-linear elastic model, presented in [35], is used to provide a relation between the pressure and the wall deformation by also involving the cross-sectional area (A). This is written as

$$p = p_{ext} + \beta(\sqrt{A} - \sqrt{A_0}) \tag{3.5}$$

where p_{ext} is the external pressure from the surrounding tissue, A_0 is the area when there is zero transmural pressure (i.e. $p = p_{ext}$) and β accounts for the material properties of the elastic vessel (A_0 also appears) and is independent of the transmural pressure,

$$\beta = \frac{\sqrt{\pi} h E}{A_0(1 - \sigma^2)} \tag{3.6}$$

where h is the wall thickness, E is Young's modulus and σ is the Poisson ratio. Note that only A_0 and h vary along the artery. A_0 is also considered as the starting artery shape in all computations. Since the system is highly coupled and non-linear, a numerical solution is required.

3.2 Formulation as a 2-Equation System

In order to reduce the number of variables in the system to be solved, one may replace the pressure in the system of governing equations with its derivative computed from eq. (3.5),

$$\frac{\partial p}{\partial x} = \frac{\partial p_{ext}}{\partial x} + \frac{\beta}{2\sqrt{A}}\frac{\partial A}{\partial x} - \frac{\beta}{2\sqrt{A_0}}\frac{\partial A_0}{\partial x} + (\sqrt{A} - \sqrt{A_0})\frac{\partial \beta}{\partial x} \tag{3.7}$$

By doing so, the 3-equation system, eqs. (3.7) and (3.4), takes the form of a quasi-linear system of 2 equations, written in non-conservative form, as follows,

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{H}\frac{\partial \mathbf{U}}{\partial x} = \mathbf{C} \tag{3.8}$$

where

$$\mathbf{U} = \begin{bmatrix} A \\ u \end{bmatrix}, \ \mathbf{H} = \begin{bmatrix} u & A \\ \frac{\beta}{2\rho\sqrt{A}} & u \end{bmatrix}$$

$$\mathbf{C} = -\frac{1}{\rho}\begin{bmatrix} 0 \\ \frac{\mu^* \rho u}{A} + \frac{\partial p_{ext}}{\partial x} - \frac{\beta}{2\sqrt{A_0}}\frac{\partial A_0}{\partial x} + (\sqrt{A} - \sqrt{A_0})\frac{\partial \beta}{\partial x} \end{bmatrix}$$

eq. (3.8) must be solved to compute the 2 unknowns (A and u) at each node.

3.3 Flux Vector Splitting Scheme

The eigenvalue vector $\boldsymbol{\Lambda}$ of the coefficient matrix $\mathbf{H}$ results by solving the equation $|\ \boldsymbol{\Lambda}\ \mathbf{I}$ - $\mathbf{H}\ | = 0$, [38] which gives:

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} u + c \\ u - c \end{bmatrix} \tag{3.9}$$

where c is the speed at which a small pulse propagates through the artery (wave speed),

$$c = \sqrt{\frac{\beta\sqrt{A}}{2\rho}} \tag{3.10}$$

Since the flow is physiological, the velocity is less than the wave speed ($u < c$) and, thus, the eigenvalues are real and the system is hyperbolic. The physical meaning of the eigenvalues is that cross-sectional area and velocity wave fronts propagate through the artery forward with velocity $\lambda_1 = u + c$ and backward with velocity $\lambda_2 = u - c$. The set of left eigenvectors is computed by solving the equation $\mathbf{I_i H} = \lambda_i \mathbf{I_i}$, where $\mathbf{I_i}$ is the right eigenvector of $\mathbf{H}$ corresponding to λ_i,

$$\boldsymbol{L^{-1}} = \begin{bmatrix} \boldsymbol{I_1^T} \\ \boldsymbol{I_2^T} \end{bmatrix} = \begin{bmatrix} c/A & 1 \\ -c/A & 1 \end{bmatrix} \tag{3.11}$$

and, therefore, matrix $\mathbf{L}$ is computed as

$$\boldsymbol{L} = \frac{A}{2c} \begin{bmatrix} 1 & -1 \\ c/A & c/A \end{bmatrix} \tag{3.12}$$

Note that $\mathbf{H}=\boldsymbol{L\Lambda L^{-1}}$. The coefficient matrix $\mathbf{H}$ is then decomposed into a positive $\mathbf{H^+}$ and a negative $\mathbf{H^-}$, such that: $\mathbf{H^+}+\mathbf{H^-}=\mathbf{H}$. Each component is defined by using the properly signed eigenvalues as follows,

$$\mathbf{H^+} = \boldsymbol{L\Lambda^+ L^{-1}} = \frac{1}{2} \begin{bmatrix} \lambda_1 & \frac{A}{c}\lambda_1 \\ \frac{c}{A}\lambda_1 & \lambda_1 \end{bmatrix} \tag{3.13}$$

$$\mathbf{H^-} = \boldsymbol{L\Lambda^- L^{-1}} = \frac{1}{2} \begin{bmatrix} \lambda_2 & -\frac{A}{c}\lambda_2 \\ -\frac{c}{A}\lambda_2 & \lambda_2 \end{bmatrix} \tag{3.14}$$

where

$$\boldsymbol{\Lambda^+} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & 0 \end{bmatrix} \text{ and } \boldsymbol{\Lambda^-} = \begin{bmatrix} 0 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

The artery length is discretized with K equidistant nodes and the total simulation time is discretized with T time instants, as well. Applying 2nd order Flux Vector Splitting to the system of equations, eq. (3.8), leads to the discretized form at time instant $n + \frac{1}{2}$ and in L-th node,

$$\frac{\partial}{\partial t}\mathbf{U}_K^{n+\frac{1}{2}} + \mathbf{H^+}\mathbf{U}_{plus}^{n+\frac{1}{2}}\frac{1}{\Delta x} + \mathbf{H^-}\mathbf{U}_{minus}^{n+\frac{1}{2}}\frac{1}{\Delta x} = \mathbf{C}_k^{n+\frac{1}{2}} \tag{3.15}$$

where the plus and minus indices represent the $\mathbf{U}$ one-sided difference stencils for $\mathbf{H}^+$ and $\mathbf{H}^-$, respectively,

$$\mathbf{U}_{plus}^{n+\frac{1}{2}} = \frac{3}{2}\mathbf{U}_k^{n+\frac{1}{2}} - 2\mathbf{U}_{k-1}^{n+\frac{1}{2}} + \frac{1}{2}\mathbf{U}_{k-2}^{n+\frac{1}{2}}$$
$$\mathbf{U}_{minus}^{n+\frac{1}{2}} = -\frac{3}{2}\mathbf{U}_k^{n+\frac{1}{2}} + 2\mathbf{U}_{k+1}^{n+\frac{1}{2}} - \frac{1}{2}\mathbf{U}_{k+2}^{n+\frac{1}{2}}$$

Exponent (n) and index (k) represent time instant and node, respectively. By replacing $\mathbf{U}_k^{n+\frac{1}{2}} = \mathbf{U}_k^n + \frac{1}{2}\Delta\mathbf{U}_k^n$ and $\mathbf{C}_k^{n+\frac{1}{2}} = \mathbf{C}_k^n + \frac{1}{2}\Delta\mathbf{C}_k^n$ in eq. (3.15), the following system emerges. Vector $\Delta\mathbf{U}^n$ makes up the matrix of independent variables,

$$\frac{1}{\Delta t}\Delta\mathbf{U}_k^n + \mathbf{H}^+\Delta\mathbf{U}_{plus}^n\frac{1}{2\Delta x} + \mathbf{H}^-\Delta\mathbf{U}_{minus}^n\frac{1}{2\Delta x} - \frac{1}{2}\Delta\mathbf{C}_k^n = \mathbf{C}_k^n - \mathbf{H}^+\mathbf{U}_{plus}^n\frac{1}{2\Delta x} - \mathbf{H}^-\mathbf{U}_{minus}^n\frac{1}{2\Delta x} \tag{3.16}$$

where

$$\Delta\mathbf{U}_{plus}^n = \frac{3}{2}\Delta\mathbf{U}_k^n - 2\Delta\mathbf{U}_{k-1}^n + \frac{1}{2}\Delta\mathbf{U}_{k-2}^n$$
$$\Delta\mathbf{U}_{minus}^n = -\frac{3}{2}\Delta\mathbf{U}_k^n + 2\Delta\mathbf{U}_{k+1}^n - \frac{1}{2}\Delta\mathbf{U}_{k+2}^n$$

and, $\mathbf{C}_k^n$ and $\Delta\mathbf{C}_k^n$ are given from

$$\mathbf{C}_k^n = -\frac{1}{\rho}\begin{bmatrix} 0 \\ \frac{\mu^*\rho u}{A} + \frac{\partial p_{ext}}{\partial x} - \frac{\beta}{2\sqrt{A_0}}\frac{\partial A_0}{\partial x} + (\sqrt{A} - \sqrt{A_0})\frac{\partial \beta}{\partial x} \end{bmatrix}_k^n$$

$$\Delta\mathbf{C}_k^n = -\frac{1}{\rho}\begin{bmatrix} 0 \\ \frac{\mu^*\rho\Delta u}{2A} - \frac{\mu^*\rho u\Delta A}{2A^2} + \frac{\Delta A}{4\sqrt{A}}\frac{\partial \beta}{\partial x} \end{bmatrix}_k^n = \mathbf{K}_k^n\Delta\mathbf{U}_k^n$$
$$\text{with } \mathbf{K}_k^n = -\frac{1}{\rho}\begin{bmatrix} 0 & 0 \\ -\frac{\mu^*\rho u}{2A^2} + \frac{1}{4\sqrt{A}}\frac{\partial \beta}{\partial x} & \frac{\mu^*\rho}{2A} \end{bmatrix}$$

The governing equations are written as a 5-diagonal block system by introducing five 2×2 coefficient matrices,

$$\mathbf{V}_k^n\Delta\mathbf{U}_{k-2}^n + \mathbf{X}_k^n\Delta\mathbf{U}_{k-1}^n + \mathbf{Y}_k^n\Delta\mathbf{U}_k^n + \mathbf{Z}_k^n\Delta\mathbf{U}_{k+1}^n + \mathbf{W}_k^n\Delta\mathbf{U}_{k+1}^n = \mathbf{RHS}_k^n \tag{3.17}$$

where the coefficient matrices are

$$\mathbf{V}_k^n = \frac{1}{2}\mathbf{H}^+\frac{1}{2\Delta x} = \frac{1}{2}\begin{bmatrix} \frac{\lambda_1}{4\Delta x} & \frac{A\lambda_1}{4c\Delta x} \\ \frac{c\lambda_1}{4A\Delta x} & \frac{\lambda_1}{4\Delta x} \end{bmatrix}_k^n$$

$$\mathbf{X}_k^n = -2\mathbf{H}^+\frac{1}{2\Delta x} = -\frac{1}{2}\begin{bmatrix} \frac{\lambda_1}{\Delta x} & \frac{A\lambda_1}{c\Delta x} \\ \frac{c\lambda_1}{A\Delta x} & \frac{\lambda_1}{\Delta x} \end{bmatrix}_k^n$$

$$\mathbf{Y}_k^n = \frac{3}{2}(\mathbf{H}^+ - \mathbf{H}^-)\frac{1}{2\Delta x} + \frac{1}{\Delta t}\mathbf{I} - \frac{1}{2}\mathbf{K} = \begin{bmatrix} \frac{3c}{4\Delta x} + \frac{1}{\Delta t} & \frac{3Au}{4c\Delta x} \\ \frac{3cu}{4A\Delta x} - \frac{\mu^* u}{2A^2} + \frac{1}{4\rho\sqrt{A}}\frac{\partial\beta}{\partial x} & \frac{3c}{4A\Delta x} + \frac{1}{\Delta t} + \frac{\mu^*}{2A} \end{bmatrix}_k^n$$

$$\mathbf{Z}_k^n = 2\mathbf{H}^-\frac{1}{2\Delta x} = \frac{1}{2}\begin{bmatrix} \frac{\lambda_2}{\Delta x} & -\frac{A\lambda_2}{c\Delta x} \\ -\frac{c\lambda_2}{A\Delta x} & \frac{\lambda_2}{\Delta x} \end{bmatrix}_k^n$$

$$\mathbf{W}_k^n = -\frac{1}{2}\mathbf{H}^+\frac{1}{2\Delta x} = \frac{1}{2}\begin{bmatrix} -\frac{\lambda_2}{4\Delta x} & \frac{A\lambda_2}{4c\Delta x} \\ \frac{c\lambda_2}{4A\Delta x} & -\frac{\lambda_2}{4\Delta x} \end{bmatrix}_k^n$$

and the RHS is

$$\mathbf{RHS}_k^n = \begin{bmatrix} -\frac{1}{2\Delta x}(A^+\lambda_1 + \frac{A\lambda_1 u^+}{c} + A^-\lambda_2 - \frac{A\lambda_2 u^-}{c}) \\ -\frac{1}{2\Delta x}(u^+\lambda_1 + \frac{c\lambda_1 A^+}{A} + u^-\lambda_2 - \frac{c\lambda_2 A^-}{A}) + C_2 \end{bmatrix}_k^n$$

where

$$A_k^{n,+} = \frac{3}{2}A_k^n - 2A_{k-1}^n + \frac{1}{2}A_{k-2}^n$$
$$u_k^{n,+} = \frac{3}{2}u_k^n - 2u_{k-1}^n + \frac{1}{2}u_{k-2}^n$$

$$A_k^{n,-} = -\frac{3}{2}A_k^n + 2A_{k+1}^n - \frac{1}{2}A_{k+2}^n$$
$$u_k^{n,-} = -\frac{3}{2}u_k^n + 2u_{k+1}^n - \frac{1}{2}u_{k+2}^n$$

$$C_{2,k}^n = -\frac{1}{\rho}(\frac{\mu^*\rho u}{A} + \frac{\partial p_{ext}}{\partial x} - \frac{\beta}{2\sqrt{A_0}}\frac{\partial A_0}{\partial x} + (\sqrt{A} - \sqrt{A_0})\frac{\partial\beta}{\partial x})\,|_k^n$$

At the second and the before-last nodes of the 1-D spatial domain, due to the absence of enough adjacent nodes on the one side (without interrupting the diagonal form of the system), the system becomes 1-st order,

$$\mathbf{X}_k^n\Delta\mathbf{U}_{k-1}^n + \mathbf{Y}_k^n\Delta\mathbf{U}_k^n + \mathbf{Z}_k^n\Delta\mathbf{U}_{k+1}^n = \mathbf{RHS}_k^n \tag{3.18}$$

and, there, the coefficient matrices are

$$\mathbf{X}_k^n = -\mathbf{H}^+ \frac{1}{2\Delta x} = -\frac{1}{2} \begin{bmatrix} \frac{\lambda_1}{2\Delta x} & \frac{A\lambda_1}{2c\Delta x} \\ \frac{c\lambda_1}{2A\Delta x} & \frac{\lambda_1}{2\Delta x} \end{bmatrix}_k^n$$

$$\mathbf{Y}_k^n = (\mathbf{H}^+ - \mathbf{H}^-)\frac{1}{2\Delta x} + \frac{1}{\Delta t}\mathbf{I} - \frac{1}{2}\mathbf{K} = \begin{bmatrix} \frac{c}{2\Delta x} + \frac{1}{\Delta t} & \frac{Au}{2c\Delta x} \\ \frac{cu}{2A\Delta x} - \frac{\mu^* u}{2A^2} + \frac{1}{4\rho\sqrt{A}}\frac{\partial \beta}{\partial x} & \frac{c}{2A\Delta x} + \frac{1}{\Delta t} + \frac{\mu^*}{2A} \end{bmatrix}_k^n$$

$$\mathbf{Z}_k^n = \mathbf{H}^- \frac{1}{2\Delta x} = \frac{1}{2} \begin{bmatrix} \frac{\lambda_2}{2\Delta x} & -\frac{A\lambda_2}{2c\Delta x} \\ -\frac{c\lambda_2}{2A\Delta x} & \frac{\lambda_2}{2\Delta x} \end{bmatrix}_k^n$$

3.4 Windkessel Model

The segment of the simulated artery is part of a complex network of arteries consisting the human cardiovascular system. In order to take the effect of arteries located beyond this segment into account, it is necessary to include a lumped-element model, which simplifies this effect. The outlet lumped parameter model, known as the Windkessel Model, [39], is used.

Large arteries are quite elastic due to their structure, which comprises elastic fibers. Thus, they are able to either distend or recoil, depending on the part of the cardiac cycle (systole/diastole). This behaviour leads to a flowrate difference between the inlet and outlet of the artery, creating a biological capacitor that temporarily stores the excess blood. This capacitor charges during the systole and discharges the blood to the peripheral arteries, during the diastole of the heart. It plays the role of a damper, resulting to a relatively smooth blood flow in the peripheral arteries, despite the pressure fluctuations presented over the cardiac cycle. This interaction is similar to a fire hose in which an air chamber (Windkessel in German) damps the pulsatile action of the pump, fig. 3.1. Note that in order for the blood to flow, it is required for this to overcome the vascular resistance of the peripheral arterioles and capillaries.

3.4.1 Two-Element Model

The mathematical model that simulates this phenomenon consists of two elements: the total arterial compliance (C_s) which accounts for the elasticity of the larger arteries and the peripheral resistance (R_2) which accounts for the resistance of the peripheral arteries.

The two-element Windkessel model, fig. 3.2, also referred as the CR_2 model, is

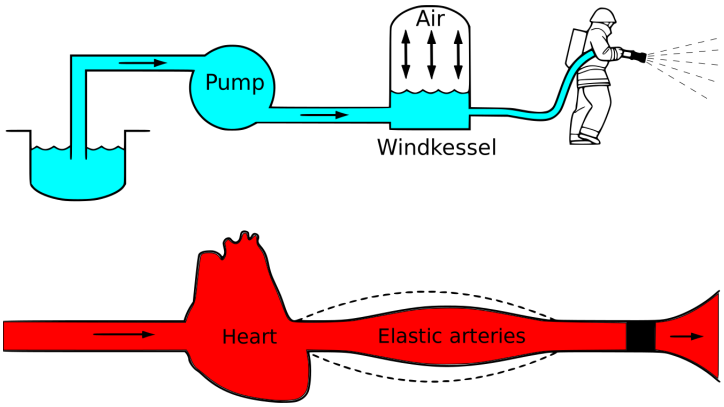


Figure 3.1: *Illustration of the Windkessel analogy. By Kurzon - Own work, CC BY-SA 3.0,* `https://commons.wikimedia.org/w/index.php?curid=31288770`

governed by a first-order differential equation,

$$\dot{Q}(t) = \frac{p(t)}{R_2} + C_s \frac{dp(t)}{dt} \tag{3.19}$$

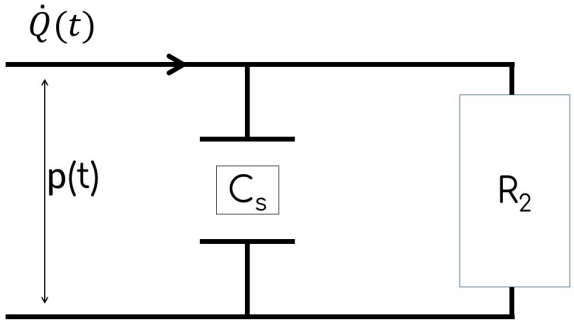


Figure 3.2: *The two-element Windkessel model. The model consists of two elements: the total arterial compliance (C_s) and the peripheral resistance (R_2).*

3.4.2 Three-Element Model

The three-element Windkessel model (R_1CR_2), fig. 3.3, is based on the two-element Windkessel model with an additional characteristic resistance (R_1) in order to eliminate abnormal reflected pulse wave oscillations. R_1 takes into account the effects (compliance and resistance) of the very proximal aorta to the simulated segment. Let p_c denote the pressure across the compliance C_s; the pressure at the end of the artery (p) is given by:

$$p(t) = p_c(t) + R_1 Q(t) \tag{3.20}$$

where p_c is given by eq. (3.19) if the pressure p is substituted with the pressure across C_s (p_c). Thus, the final system of the Windkessel model equations is derived

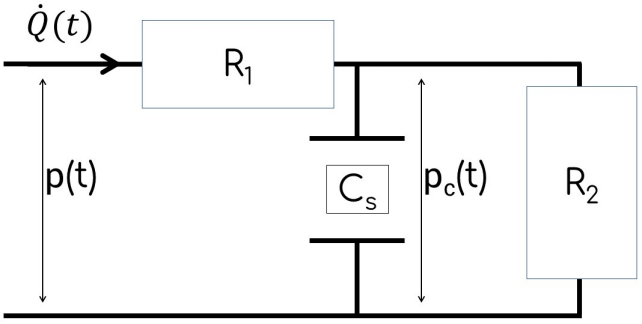


Figure 3.3: *The three-element Windkessel model. The model is based on the two-element model with an additional characteristic resistance (R_1).*

providing the pressure value at the last node (p_K),

$$p(t) = p_c(t) + R_1\dot{Q}(t) \tag{3.21}$$

$$\dot{Q}(t) = \frac{p_c(t)}{R_2} + C_s\frac{dp_c(t)}{dt} \tag{3.22}$$

In this study, the three-element Windkessel model is used to update the pressure at the last node of the artery (K) using the $\dot{Q}_{inlet}$ as $\dot{Q}$.

3.4.3 Numerical Solution

The system of eqs. (3.21) and (3.22) is numerically solved. If $\dot{Q}^{n-\frac{1}{2}}$ is given by $\dot{Q}^{n-\frac{1}{2}} = \frac{1}{2}(\dot{Q}^{n-1} + \dot{Q}^{n})$, eq. (3.22) can be discretized as,

$$\begin{aligned} \dot{Q}^{n-\frac{1}{2}} = C_s\frac{p_c^n - p_c^{n-1}}{\Delta t} + \frac{p_c^n + p_c^{n-1}}{2R_2} \Longrightarrow \\ p_c^n = \frac{(\dot{Q}^{n-\frac{1}{2}} - (\frac{1}{2R_2} - \frac{C_s}{\Delta t}))}{\frac{C_s}{\Delta t} + \frac{1}{2R_2}} \end{aligned} \tag{3.23}$$

Therefore, the pressure at the last node of the artery, at the n-th time instant, is given by

$$p_K^n = p_c^n + R_1\dot{Q}^n \tag{3.24}$$

3.5 Initial & Boundary Conditions

The cross-sectional area distribution is initialized by setting it to the A_0 distribution. The initial velocity distribution is constant and computed from the equation $\dot{Q} = Au$, where A and $\dot{Q}$ take on the corresponding initial values. The initial pressure distribution is equal to p_{ext}.

For the inlet boundary condition (k=1):
Cross-sectional Area: Since the instantaneous flowrate ($\dot{Q}^n$) is known, the velocity at the (n+1)-th time instant may be computed from the continuity equation $\dot{Q} = Au$,

$$u^{n+1} = u^n + \Delta u^n = \frac{\dot{Q}_1^{n+1}}{A_1^{n+1}}$$

if $\frac{1}{A_1^{n+1}}$ is linearized as $\frac{1}{A_1^n} - \frac{\Delta A_1^n}{(A_1^n)^2}$, then

$$\frac{\dot{Q}_1^{n+1}}{(A_1^n)^2}\Delta A_1^n + \Delta u_1^n = \frac{\dot{Q}_1^{n+1}}{A_1^n} - u_1^n \tag{3.25}$$

Velocity: A first-order Neumann boundary condition is imposed, where $\frac{\partial u}{\partial x}{}_1^n = 0$. Thus, $u_1^n = u_2^n$ and,

$$\Delta u_1^n = \Delta u_2^n \tag{3.26}$$

For the outlet boundary condition (k=K):
Cross-sectional Area: Since the pressure at the last node (p_K) is computed by the Windkessel model, eq. (3.24), the area can be computed as

$$A_K^{n+1} = (\frac{p_K^{n+1} - p_{ext}}{\beta_K} + \sqrt{A_{0,K}})^2 \overset{A_K^{n+1} = A_K^n + \Delta A_K^n}{\Longrightarrow}$$
$$\Delta A_K^n = (\frac{p_K^{n+1} - p_{ext}}{\beta_K} + \sqrt{A_{0,K}})^2 - A_K^n \tag{3.27}$$

Velocity: A Neumann condition is imposed, where $\frac{\partial u}{\partial x}{}_K^n = 0$, as well. Therefore, $u_K^n = u_{K-1}^n$ and,

$$\Delta u_K = \Delta u_{K-1} \tag{3.28}$$

or in matrix form:

Inlet:

$$\mathbf{Y}_1^n \Delta \mathbf{U}_1^n + \mathbf{Z}_1^n \Delta \mathbf{U}_2^n = \mathbf{RHS}_1^n \tag{3.29}$$

$$\mathbf{Y}_1^n = \begin{bmatrix} 1 & 0 \\ \frac{\dot{Q}_1^{n+1}}{A_1^{2^n}} & 1 \end{bmatrix}$$
$$\mathbf{Z}_1^n = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}$$
$$\mathbf{RHS}_1^n = \begin{bmatrix} 0 \\ \frac{\dot{Q}_1^{n+1}}{A_1^n} - u_1^n \end{bmatrix}$$

Outlet:

$$\mathbf{X}_K^n \Delta \mathbf{U}_{K-1}^n + \mathbf{Y}_K^n \Delta \mathbf{U}_K^n = \mathbf{RHS}_K^n \tag{3.30}$$

$$\mathbf{X}_K^n = \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix}$$
$$\mathbf{Y}_K^n = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
$$\mathbf{RHS}_K^n = \begin{bmatrix} 0 \\ (\frac{p_K^{n+1} - p_{ext}}{\beta_K} + \sqrt{A_{0,K}})^2 - A_K^n \end{bmatrix}$$

3.6 The 1DAS Software

The above flow model is programmed as a quasi-1D flow solver software (to be referred as 1DAS). Code programming was carried out in FORTRAN while data pre- and post-processing in Python 3.6.
The 1DAS software inputs comprise the parameters of the previously presented equations and consist of:

- The number of equidistant spatial nodes (K).
- The number of time instants per period ($kdit$).
- The total simulation time (T_{tot}).

- The length of the artery (L_{art}).
- The starting shape of the artery ($A_0(x)$).
- The time dependent blood inflow ($\dot{Q}_{inlet}(t)$).
- Blood Properties:
  - Density (ρ_b)
  - Kinematic Viscosity (ν_b)
  - Pressure in the surrounding tissue (p_{ext})
- Artery Properties:
  - Young Modulus (E)
  - Poisson Ratio (σ)
  - Spatial distribution of thickness ($h(x)$)
- Windkessel model parameters:
  - Initial Pressure of Capacitor (p_c)
  - Capacitance (C_s)
  - Resistance R1 (R_1)
  - Resistance R2 (R_2)

The software solves the 5-diagonal block linear system with the block elimination method. This method effectively computes the LU factorization of the coefficient matrices, then, performs the forward block substitution and, at the end, the solution emerges through the backward substitution. It is a modified version (two additional coefficient matrices) of the 3-diagonal block linear system solver, called *LU Factorization*, which is presented in [40]. This method was already programmed as a FORTRAN subroutine and is utilized by the 1DAS as its system solver. It is worth noting the creation of a PYTHON code, which is capable of plotting 1DAS software results while running. This code assisted the 1DAS software development and live results evaluation.

3.6.1 Examples of 1DAS runs

In order to showcase the running of the 1DAS software, a run of a dummy case was performed. The distributions of the initial cross-sectional area and thickness are shown in fig. 3.4. A physiological blood flow is used by the software, fig. 3.4. How these curves were generated and their physical meaning are discussed in Chapter 5.

The fig. 3.5 shows the 3-D plots of the cross-sectional area and velocity distributions for one period (after the solution is converged). The results showcase the capabilities

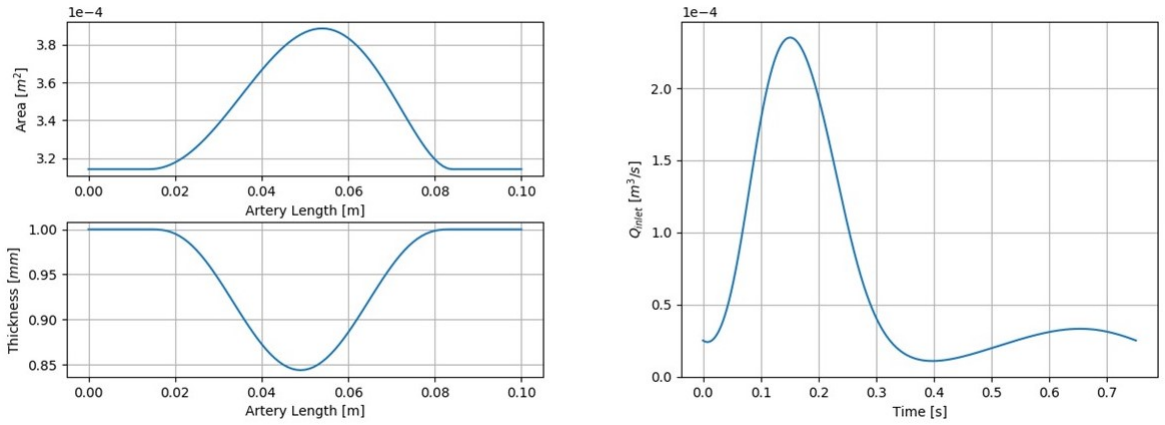


Figure 3.4: *The cross-sectional area and thickness distributions (left) used by the 1DAS software for the dummy case. The physiological time-varying bloodflow (right).*

of the software. The elastic walls of the artery adapt to the evolving flowrate and pressure. The inlet velocity profile follows the bloodflow form.

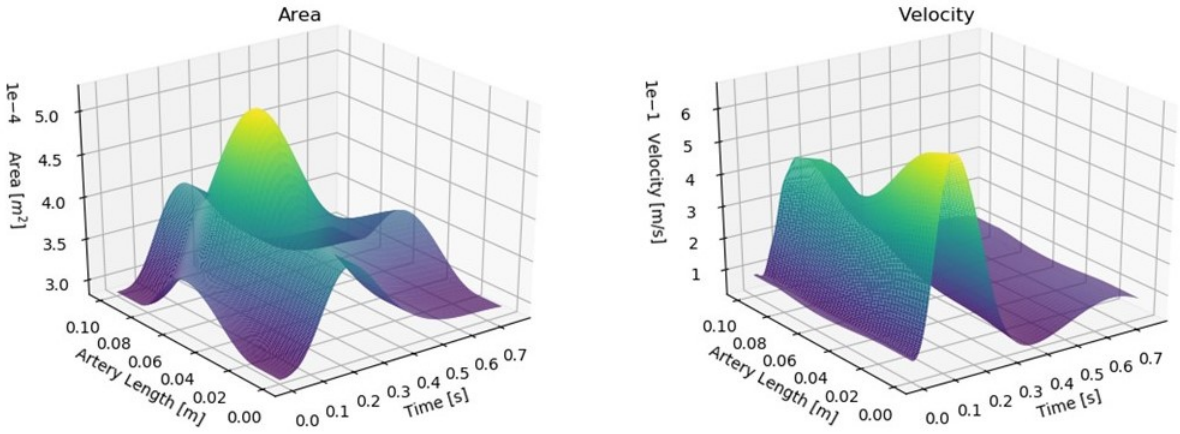


Figure 3.5: *The cross-sectional area and velocity distributions generated by the 1DAS software for the dummy case.*

3.7 Artery with Rigid Walls

For performing the first tests, a version of 1DAS to be referred as 1DASR, was created, but with an artery with rigid walls. The flow is governed by a system of equations,

$$\frac{d(Au)}{dx} = 0 \tag{3.31}$$

$$\frac{du}{dt} + u\frac{du}{dx} + \frac{dp}{dx} = -\frac{\mu^* u}{A} \tag{3.32}$$

The velocity is given from the continuity equation, where cross-sectional area (A) is varied along the artery and the bloodflow ($\dot{Q}$) is time-dependent,

$$u(x,t) = \frac{\dot{Q}(t)}{A(x)} \tag{3.33}$$

and, thus, by taking the derivatives of it,

$$\frac{du}{dx} = -\frac{\dot{Q}}{A^2}\frac{dA}{dx} \tag{3.34}$$

$$\frac{du}{dt} = -\frac{1}{A}\frac{d\dot{Q}}{dt} \tag{3.35}$$

The artery length is discretized with K equidistant nodes and the total simulation time is discretized with T time instants, as well. By replacing the velocity derivatives in eq. (3.32), and discretizing the equation, the following formula is derived for time-step n and in node k,

$$\frac{dp}{dx}^n_k = -\frac{\mu^* \dot{Q}^n}{A_k^2} + \frac{(\dot{Q}^n)^2}{A_k^3}\frac{dA}{dx}_k - \frac{1}{A_k}\frac{d\dot{Q}}{dt}^n \tag{3.36}$$

The pressure derivative can be written as

$$\frac{p_{k+1}^n - p_k^n}{\Delta x} = \frac{1}{2}\left(\frac{dp}{dx}^n_k + \frac{dp}{dx}^n_{k+1}\right) \tag{3.37}$$

The pressure is given by

$$p_k^n = p_{k+1}^n - \frac{\Delta x}{2}\left(\frac{dp}{dx}^n_k + \frac{dp}{dx}^n_{k+1}\right) \tag{3.38}$$

The velocity can be computed from the discretized form of eq. (3.33),

$$u_k^n = \frac{\dot{Q}^n}{A_k} \tag{3.39}$$

Note that the pressure in the last node (K) is given from the Windkessel model, as

described in Subsection 3.4.3. The numerical solution of the discretized eqs. (3.36), (3.38) and (3.39) was programmed in FORTRAN code and its results were used for training the LSTM network.

Chapter 4

LSTM Benchmark Cases

Before moving into the main implementation of the LSTM network in biological flows, two benchmark cases were performed. Since the generation of dataset (i.e. biological flows) is time-consuming, the benchmark cases act as introduction on how the LSTM are implemented (computationally), and further exploited in predicting sequential data.

4.1 Periodic Function

Firstly, a benchmark case is introduced for exploring the capabilities of LSTM networks in prediction of periodic functions. LSTM networks are known for their performance in reconstructing time-series and memorizing patterns through time-dependent data.

A periodic function was used for generating training patterns,

$$y = a_1 sin(a_2 x) + a_3 cos(a_4 x) + sin(a_5 x) + a_6 \tag{4.1}$$

where $x \in [0, 160]$ represents the pseudo-time and $a_1, ..., a_6 \in [0, 1]$ are parameters. By varying the parameters, different time-series were generated. The period and amplitude are influenced by this variation. The x-axis was discretized using 401 equidistant nodes.

The goal of the LSTM network was to predict the following value of this function, each time, by presenting it with n_{steps} preceding instantaneous values of it. These preceding values can either be computed from eq. (4.1) or be network predictions. The training dataset comprised 28 time-series, fig. 4.1, generated by varying ran-

domly the parameters in the aforementioned space. The total computational cost for the generation of the training patterns was $\sim 5mins$.

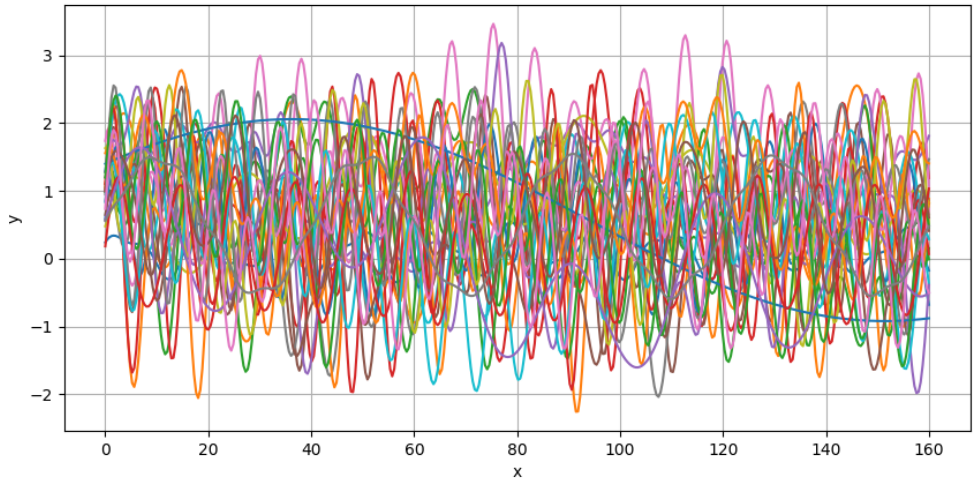


Figure 4.1: *The 28 times-series constituting the training dataset.*

The network architecture consists of 5 layers, of which the 2 first layers are the LSTM units, the following layer is the flatten layer and the last layers are the dense ones, fig. 4.2. The 2 LSTM units have 64 neurons each, and the dense have 64 neurons each, as well. The tanh activation function was used for all layers, except the last one, in which the linear activation function was utilized.

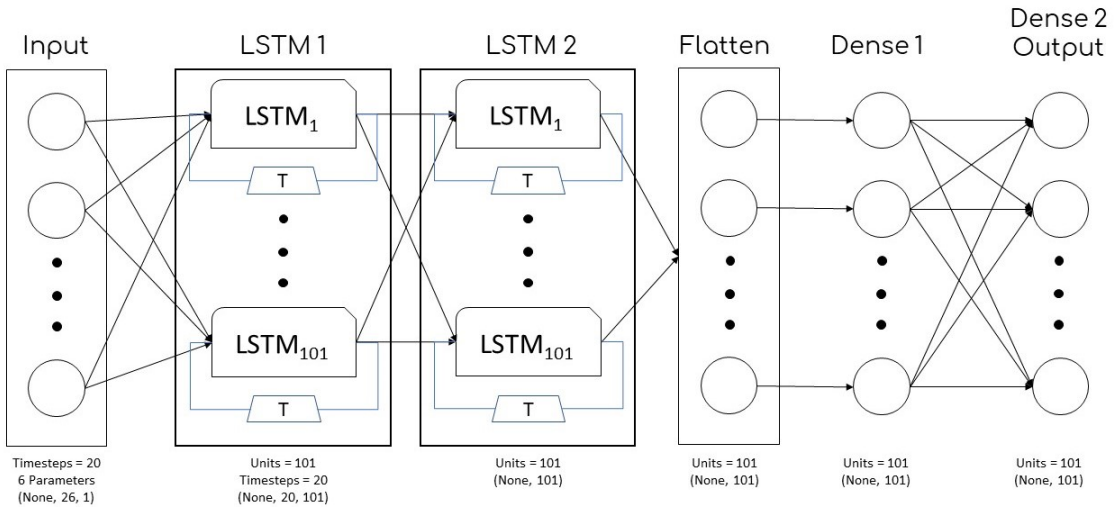


Figure 4.2: *The LSTM network architecture. The network consists of the LSTM part, a flatten layer and the dense part.*

The network was, then, trained for predicting the following value by presenting it with the 20 preceding values (n_{steps}) and the 6 values of the parameters. At first,

the network was trained for 150 epochs. It was called, then, to make predictions by feeding back its predictions to itself and utilizing them as input for the subsequent time steps. The input dataset was updated with these predictions and the network was retrained for 150 epochs. The goal of the network was to reconstruct the entire time-series, which consisted of 381 values. This number was derived by subtracting the n_{steps} from the number of equidistant nodes (401), that the x-space was discretized with. The training cost was $\sim 5mins$, on a NVIDIA 1050 GPU.

Two, not seen by the network, time-series were used for assessing its performance, fig. 4.3. The average MAE of the LSTM predictions is 3×10^{-2}. Firstly, the network was presented with 20 instantaneous values, computed from the eq. (4.1). The network predicted the following value and fed it back to itself and, used it as input for the next one. The procedure of feeding back the prediction continued until the network had reconstructed the entire time-series. The prediction for the entire time-series (381 values) is based on the 20 starting values (n_{steps}) and the 6 values of the parameters. A further explanation of both training and prediction techniques is given in Section 5.2.4. The results are more than satisfactory and indicate that LSTM networks can be utilized to predict periodic functions.

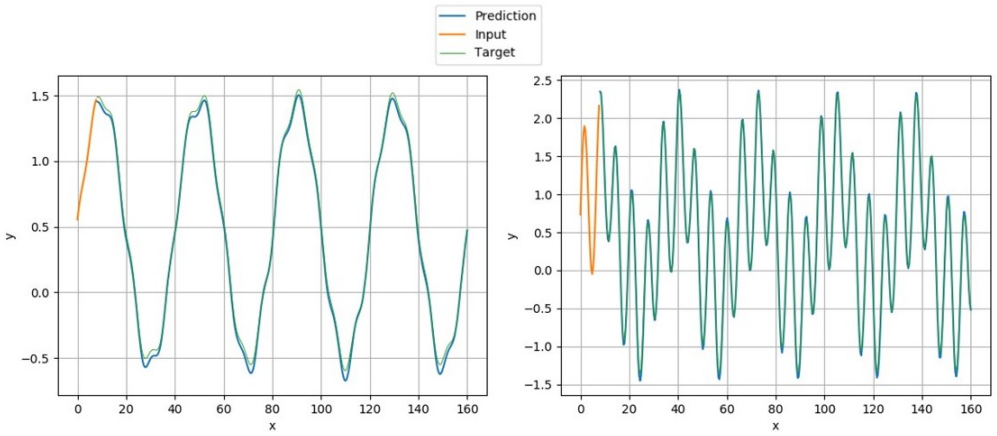


Figure 4.3: *Two, not seen by the network, time-series, reconstructed by the LSTM network.*

4.2 Heat Conduction

In the second benchmark case, the LSTM was trained to predict time-varying temperature distributions on a 1D plate, with length $L_p = 0.3m$, fig. 4.4. On the left end of the plate, there was air with time-varying temperature and, on the right end, the temperature of air was constant. This was a simulation of a metallic wall, which was adjacent to the air both inside and outside of it. The right end of the plate represented the inside (T_{in}), with the constant temperature, while the outside varied along the day (T_{out}). This phenomenon is governed by a partial differential

equation, the heat conduction equation,

$$\frac{\partial}{\partial x}(k\frac{\partial T}{\partial x}) = \rho C_{pl}\frac{\partial T}{\partial t} \tag{4.2}$$

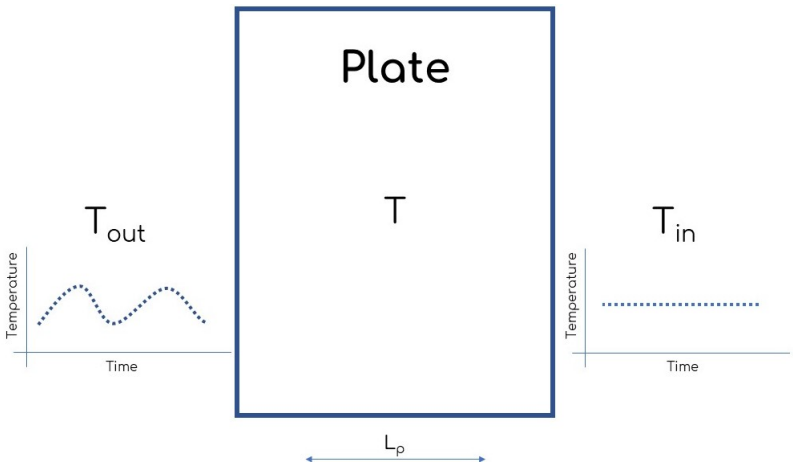


Figure 4.4: *The plate, in 2-D perspective, with constant internal and time-varying external temperatures.*

where T is the temperature, $k = 2\ W/m°C$ is the coefficient of thermal conductivity, $\rho = 2000\ kg/m^3$ is density, and $C_{pl} = 1000\ J/kg°C$ is the thermal capacity. All quantities refer to the simulated plate. At the right end, the plate was adjacent to air with temperature $T_{in} = 27°C$. At the left end, the air temperature (T_{out}) was parameterized with the following periodic function,

$$T_{out} = 35 + b_1 cos(\frac{2\pi(t - 3600b_2)}{3600b_3}) \tag{4.3}$$

where $b_1, b_2 \in [0, 24]$ and $b_3 \in [-5, 5]$ are the parameters. The convective heat transfer coefficient of the left end was $h_{out} = 28\ W/m^2K$ and on the right end $h_{in} = 8\ W/m^2K$. Eq. (4.2) was solved numerically in PYTHON. The plate width (L_{pl}) was discretized using 60 equidistant nodes (59 *cells*). The total simulation time was 3 days or $\sim 259K$ seconds and was discretized with 301 time steps, at constant time intervals. The equation was solved with the finite volumes method (cell-centered), which resulted in a tri-diagonal system of equations. The Neumann boundary conditions, at both ends, were set through the heat convection, between the solid plate and the air. The temperature of $20°C$ was used as the initialization for the temperature distribution.

By varying the parameters of the outside temperature, different temperature time-series were generated and were, later, used from the solver for determining the boundary conditions, fig. 4.5. 28 time-series were generated and, thus, 28×301

time-varying temperature distributions resulted. The solver took $\sim 1.5mins$ for generating the distributions. As shown in fig. 4.6, the solution had a transient phase that converges to a periodic solution.

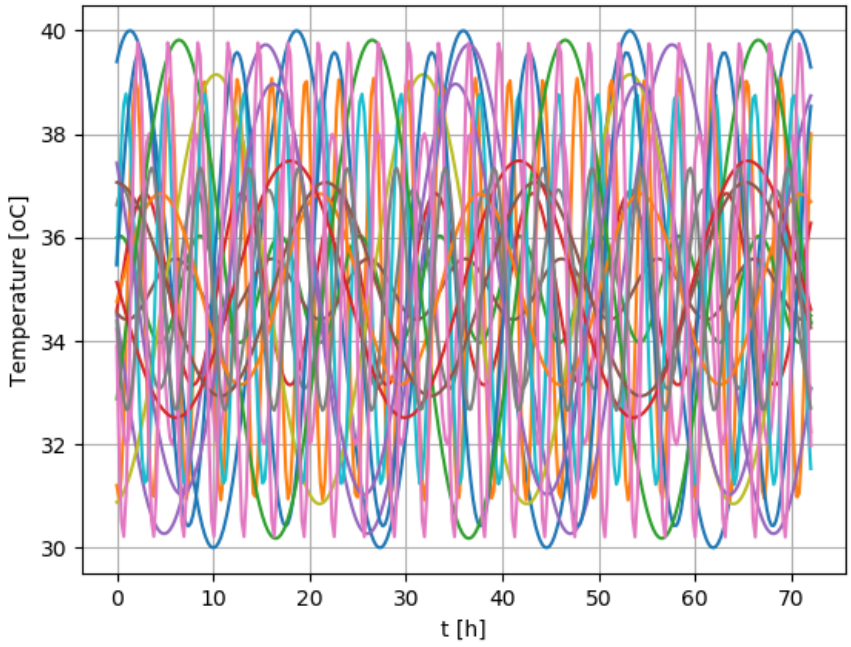


Figure 4.5: *The T_{out} time-series used for determining the boundary conditions.*

20 distributions were presented to the network in order to predict the following one and, thus, n_{steps} was equal to 20. The network architecture of fig. 4.2 was adjusted to this benchmark case. The number of hidden layers, as well as the number of their neurons, remained unchanged. The input size was (20,59) while the output size was a distribution of 59 temperatures. Note that 59 was the number of cells and therefore the number of temperature values along the plate. The training procedure was identical to the previous case. The training cost was $\sim 5mins$ on a NVIDIA 1050 GPU. Note that, the network was not presented with any information about the T_{out} (its time-series or the values of the parameters).

The network, during the prediction of the temperature distributions, was feeding back its prediction to itself, as mentioned in the previous case. The predictions of two, not seen by the network, patterns are shown in figs. 4.7 and 4.8. The average MAE of the LSTM predictions is $15 \times 10^{-2}\ ^oC$. They indicate that the LSTM network is capable of reconstructing the temperature time-series of the entire plate with the information of the transient phase, since the first 20 distributions, used by the network, are of this phase. The network used only these 20 distributions to predict the entire time-varying temperature field.

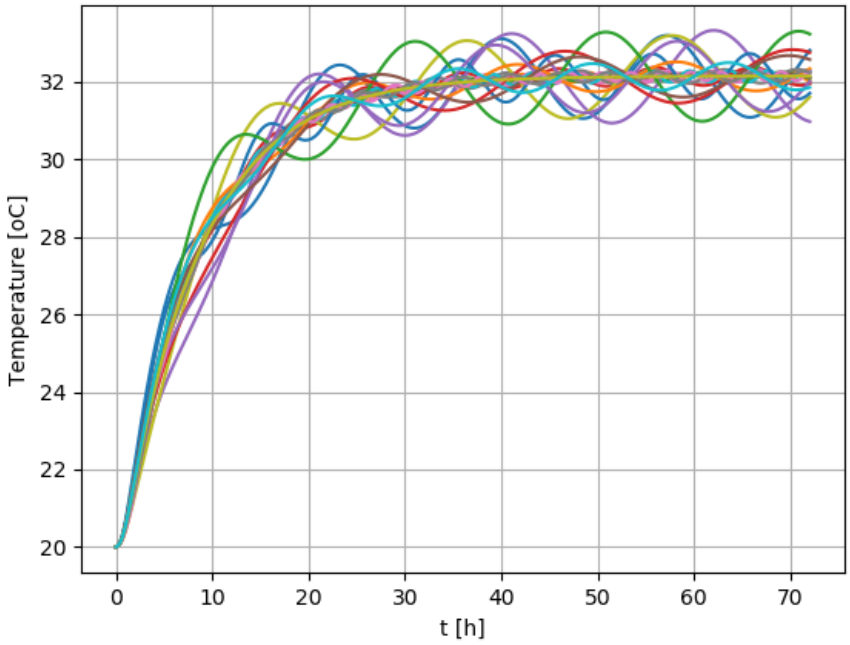


Figure 4.6: *The temperature time-series (training patterns) of the probe (30th) cell.*

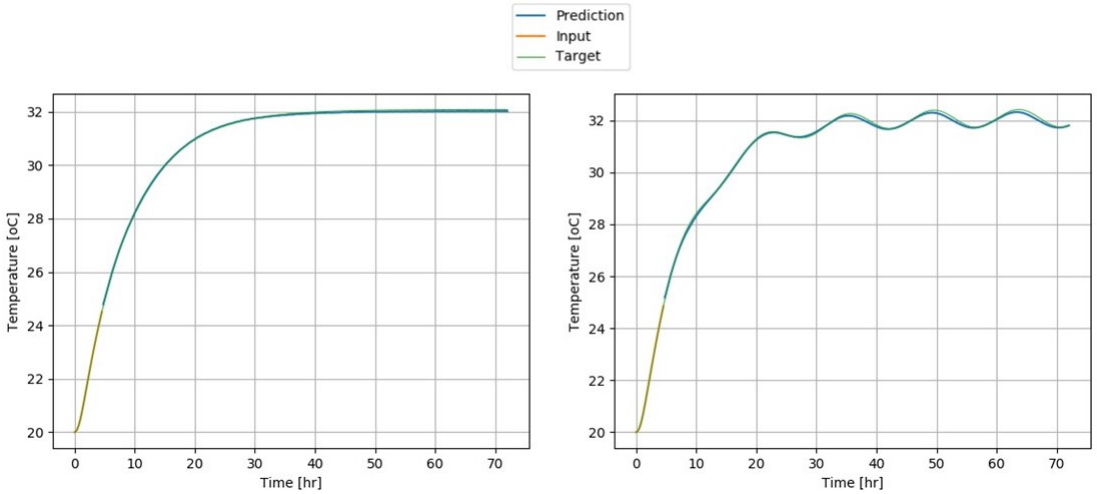


Figure 4.7: *The prediction of two, not seen by the network, temperature time-series of the probe (30th) cell.*

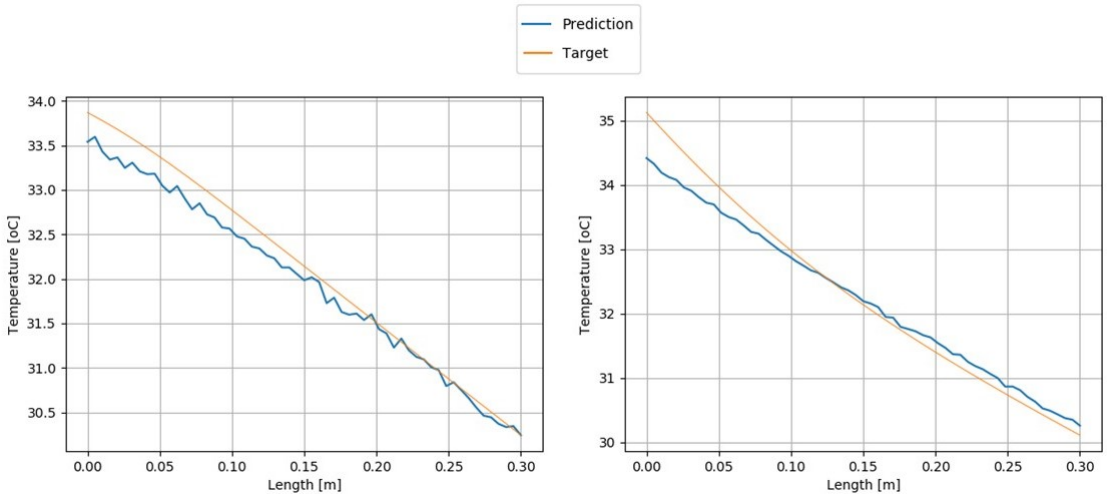


Figure 4.8: *The prediction of two, not seen by the network, temperature spatial distributions of the probe (250th or* ~ 58 *hr) time-instant.*

Chapter 5

Prediction of 1D Time-Varying Flows in Arteries with LSTM Networks

5.1 Introduction

Inhere, a code for simulating arterial flows was developed and used to generate quasi-1D flows in axisymmetric arteries. A Long Short-Term Memory (LSTM) network was trained to predict velocity distributions. It constitutes an initial study on how LSTM networks can further be used in biological flows studies, utilizing data generated from CFD software solving 3D flows. Note that the problem of realistic arterial flows is an unsteady multi-discipline problem, since the elastic walls interact with the blood, which follows the pulsatile cardiac action. Thus, the generation of 3D data for training the LSTM network would be computationally expensive.

5.2 Varying Initial Artery Shapes

The LSTM neural network was trained to reconstruct the time-series of blood flow quantities of a quasi-1D flow problem in an artery with elastic walls of a simplified axisymmetric geometry though. The network was able to predict the time-evolution of velocity and area distributions along the artery, using the corresponding distributions at previous time instants. These previous spatial distributions were either

generated by a CFD solver or, after the first time instants, they were flow predictions by the network itself.

The quasi-1D flow solver (1DAS) in arteries with flexible walls was the analysis tool providing the training and validation patterns for the network. The artery was modelled as a tube with a non-constant cross-sectional area along its length. The solver data comprise the starting cross-sectional area distribution (A_0), the wall thickness distribution, the physiological data of the artery (blood properties, artery mechanical properties and Windkessel model parameters), and the time-dependent blood inflow as boundary condition ($\dot{Q}_{inlet}$). In order to represent the pulsatile cardiac output (heart rate and stroke volume), a periodic functional form for the inflow was used. The physiological data remained constant in all runs, whereas the starting cross-sectional area distribution, the wall thickness distribution and the time-series of the blood inflow varied. In this first case, only the starting cross-sectional area distribution varied. The velocity, pressure and area distributions along the length of the blood vessel at each time instant are the output of the 1DAS software and these were used for training and, then, assessing the LSTM network.

5.2.1 Initial Cross-Sectional Area

The initial cross-sectional area distribution of the artery was arbitrarily selected and parameterized using a Bezier curve with 7 control points (CPs), fig. 5.1. Then, by varying their coordinates (with fixed abscissas), various shapes of arteries were generated and later used by the 1DAS software as the initial shape.

A constant cross-sectional area curve, computed for an artery with radius equal to $R_{art} = 0.010\ m$, was utilized as a starting artery for the parameterization procedure. The first and last CPs were fixed at 15% and 85% of the artery length. The abscissas of the other control points remained constant and distributed evenly. Note that the second and the before last ones had the same ordinate as the first and last CPs respectively, for derivative continuity purposes. By varying R_{art} by $\pm 50\%$ at each end of the artery, new first and last control CPs emerged while the ordinates of the 3 intermediate points were distributed evenly between them. The ordinates of the 3 intermediate CPs were multiplied by the factors $1.1, 1.35$ & 1.4 respectively. These factors varied from -10% to $+50\%$. In an effort to evaluate the solutions generated by the 1DAS software, an artery with inlet and outlet radii both equal to $R_{art} = 0.010m$ and factors $1.1, 1.35$ & 1.4 was selected as the reference artery, fig. 5.1.

By varying the aforementioned parameters (inlet and outlet radii & factors), a dataset of 30 arteries was generated, and these were used in the unsteady runs of the 1DAS software as initial cross-sectional area distributions, fig. 5.2.

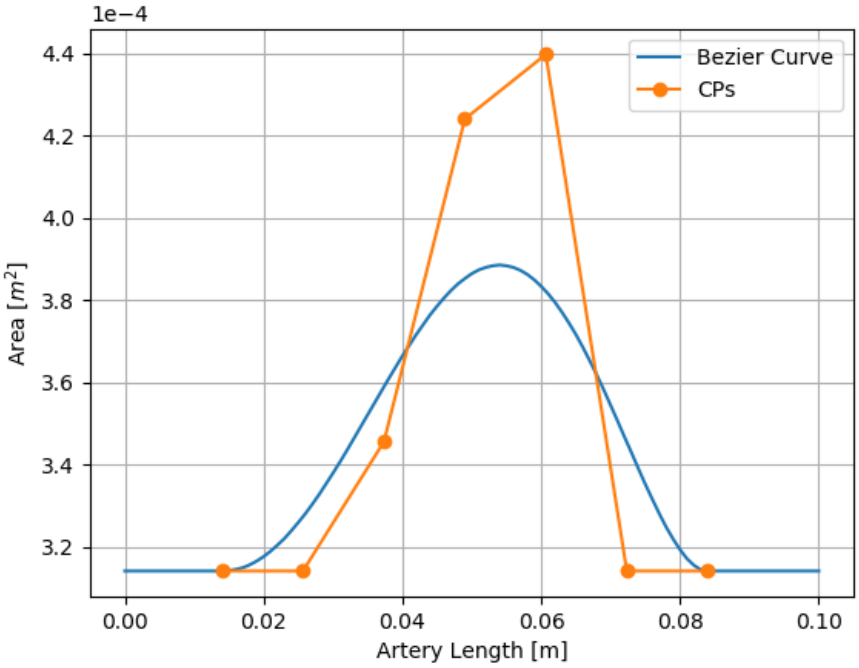


Figure 5.1: *Reference Artery. It is parameterized using a Bezier curve (blue) with 7 CPs (orange/dotted). The inlet and outlet radii are both equal to* $R_{art} = 0.010m$*, corresponding to area equal to* $3.1413 \cdot 10^{-4} m^2$*, and the intermediate CPs factors are* $1.1, 1.35$ *&* 1.4*.*

5.2.2 Blood Inflow and 1DAS Parameters

The $\dot{Q}_{inlet}$ that was used for the first case study was a very simple periodic (sinusoidal) function, fig. 5.3, with period $T_h = 0.75s$, representing a pseudo-pulsatile cardiac output, namely:

$$Q_{inlet} = 7 \cdot 10^{-5} + 5 \cdot 10^{-5} \sin \omega t \tag{5.1}$$

where $\omega = \pi / 0.75$ and $t \in [0, 0.75]$. Every quantity was written in *SI* base units.

The rest of the 1DAS software parameters are presented in table 5.1.

5.2.3 Training Dataset Creation

The total simulation time was $8s$ (corresponding to 10.6 periods of 0.75s each), with 7500 time instants in each period, resulting in 80000 time instants in total. The artery length (L_{art}) was $0.1m$ and was discretized using 101 equidistant nodes.

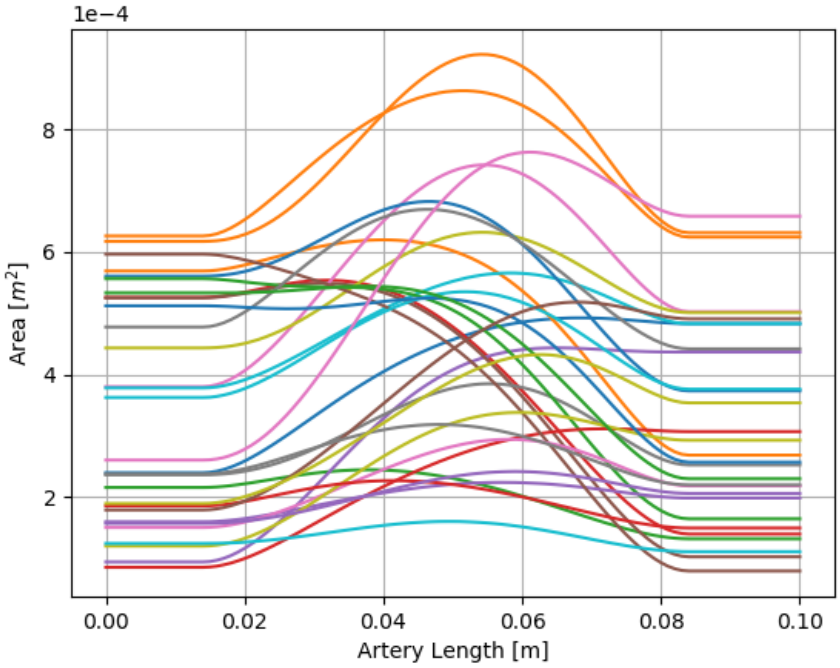


Figure 5.2: *Cross-sectional area distributions, created by varying the CPs of the Bezier curve used by the 1DAS software as initial distributions (A_0) for the 30 training patterns.*

Quantity	Symbol	Value
Blood Properties		
Density $[kg/m^3]$ [41]	ρ_b	1060
Kimematic Viscosity $[m^3/s]$ [42]	ν_b	$3.5 \cdot 10^{-6}$
External pressure $[Pa]$	P_{ext}	10^4
Artery Mechanical Properties		
Young Modulus $[Pa]$	E	$91 \cdot 10^4$
Poisson Ratio	σ	0.5
Thickness $[m]$	h	10^{-3}
Windkessel Parameters		
Initial Pressure of Capacitor $[Pa]$	p_c	$0.1 \cdot 10^4$
Capacitance $[m^4 s^2 kg^{-1}]$	C_s	$4.5 \cdot 10^{-9}$
Resistance R1 $[kgm^{-4}s^{-1}]$	R_1	$0.5 \cdot 10^7$
Resistance R2 $[kgm^{-4}s^{-1}]$	R_2	$1.8 \cdot 10^8$

Table 5.1: *Physiological data of the arteries used by the 1DAS software for all runs.*

For the time slot that corresponds to the first 6 periods ($4.5s$), $\dot{Q}_{inlet}$ was constant ($\dot{Q}_{inlet}= 0.7 \cdot 10^{-4} m^3/s$) resulting in an initial transient phase during which the flexible walls adapted themselves to the evolving pressure within the artery and, at

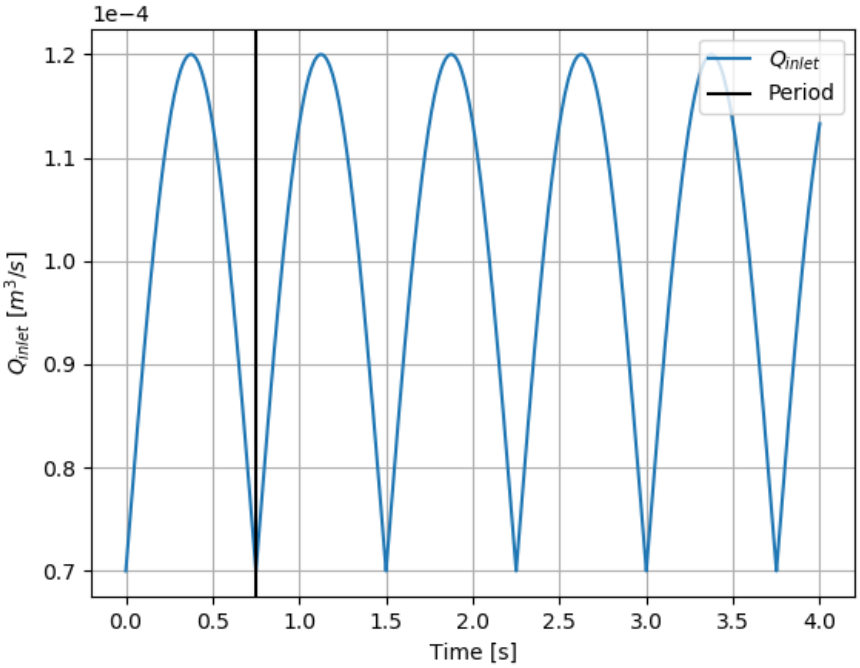


Figure 5.3: *The periodic blood inflow ($\dot{Q}_{inlet}$) of eq.* (5.1) *(blue) with period $T_h = 0.75s$ (the black vertical line delineates to the end of the first period) plotted for several periods. At both ends of each period the inflow derivative is discontinuous.*

the end, a steady flow solution was computed. The second phase started upon the end of the first one and, in this phase, the unsteady inlet flowrate profile $\dot{Q}_{inlet}$, eq. (5.1), was imposed. Splitting into two phases was decided as it was quite helpful to have the flow quantities with the steady inlet condition converged at first and, then, continue the problem solution with the time-varying flowrate. The $\dot{Q}_{inlet}$ time-series and the corresponding velocity time-series of the reference initial cross-sectional area distribution, at the 50th node (almost mid of the artery), are shown in figs.5.4 and 5.5 respectively.

The computed distributions at the selected probe node (the $50th$ node) of the corresponding cross-sectional area distributions, fig. 5.2 solved by the 1DAS software are shown in figs. 5.6, 5.7 and 5.8. In fig. 5.9, the converged area distribution at the end of the first phase is shown in comparison to the reference A_0, pointing out that the artery, at the end of the first phase, is outward bulged.

In order to reduce the memory requirements and the training cost, the distributions were sampled every 20 time instants. Practically, the training dataset was formed by distributions in every 20 time instants, skipping the intermediate ones. One distribution used by the network carries out the information of 20 distributions. Thus, each period corresponded to 375 distributions.

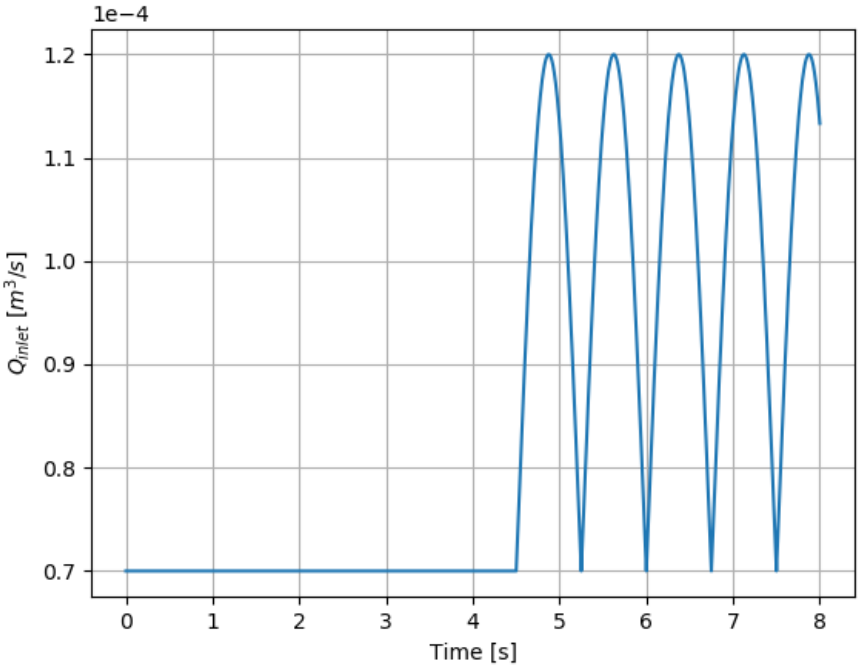


Figure 5.4: *The blood inflow ($\dot{Q}_{inlet}$) remains constant during the first 6 periods and then starts varying according to the periodic formula of eq.* (5.1) *(blue).*

5.2.4 LSTM Architecture and Training

The input of the LSTM network was a number (n_{steps}) of preceding instantaneous flow solutions. In this study, velocity longitudinal distributions were selected as the network's output using n_{steps}= 150. So, the network was presented with the 150 previous velocity distributions along the length of the artery and its role was to predict the next longitudinal distribution of the same quantity. The 150 previous time instant distributions can either be computed by the 1DAS software or be predictions of the LSTM itself. The second feature provided the network the ability to predict a whole time-series of the velocity longitudinal distributions given only an initial set of 150 distributions, by feeding back its predictions to itself.

The procedure during which the network predicts a flow quantity time-series, over the artery, was called *Prediction Procedure*. This procedure was initialized by setting 150 distributions of a selected quantity, generated by the 1DAS software, as network input. The network, then, predicted the next distribution and used it as input for the next time instant prediction, fig. 5.10. It continued to feed back its own predictions as input and after 149 time instants the network became independent of the initial 150 distributions computed by the 1DAS software , fig. 5.11. Practically, the network required only 150 distributions, as initialization, to predict a whole time-series.

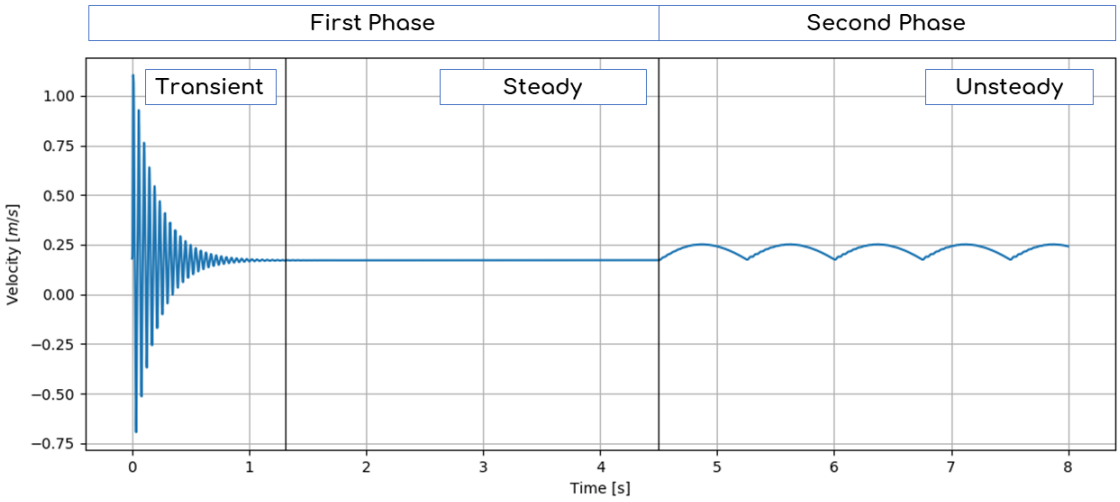


Figure 5.5: *Velocity time-series of the reference A_0 at the 50th node. The solution is divided into* 3 *parts (black vertical lines correspond to the end of each part) and two phases determined by the $\dot{Q}_{inlet}$. Severe fluctuations are presented in the beginning of the first phase.*

For the network training only 18 arteries were used plus 2 for the test set, making up 20 arteries in total. Note that not all arteries of the original dataset of 30 arteries were used due to the high resources demands of the network. More training patterns resulted in more memory demands and exceeding of the RAM limitations. The 1DAS software took $\sim$ 2.5 *mins* to solve the flow, and, thus, the total computational cost was $\sim$ 50 *mins*, for the 20 flows. This cost remained constant for all cases. During its training, the network was presented with distributions starting from $4.125s$ (half period before the end of the first phase) in order to filter out the fluctuations presented in the beginning of the first phase of the solution, as shown in fig. 5.5. The goal of the case study was to provide the network with a set of 150 distributions of the steady part of the first phase and call it to predict the unsteady one by continuing on its own and utilizing its own predictions.

The network architecture consists of 2 parts combined in the integrated LSTM network, fig. 5.12. The first part consists of LSTM cells with 2 layers of 101 neurons each while the second one consists of 2 dense layers of 101 neurons. The first LSTM cell and the last dense layer stand for the input and the output layers respectively. The 2 parts are interconnected with a flatten layer and the activation function for all layers, except the output, is the LeakyReLU, [43]. This activation function is preferably used due to its efficiency during the training procedure. In the output dense layer, the linear activation function is used. In addition, the selection of the number of layers and neurons was done by the trial and error method. Further

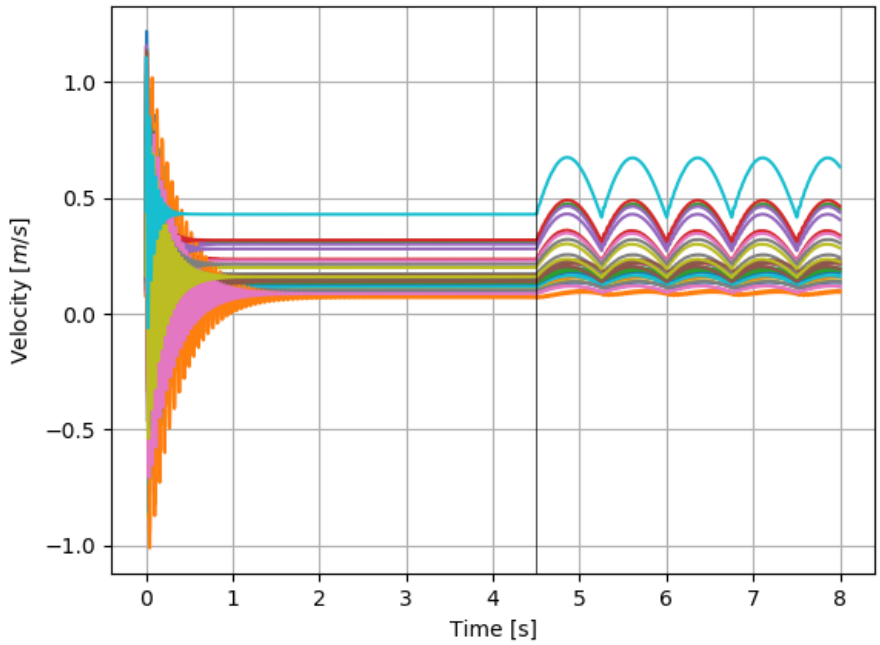


Figure 5.6: *Velocity time-series at the probe (50th) node. For the first* 4.5*s (black vertical line corresponds to the beginning of the second phase)* $\dot{Q}_{inlet}$ *is constant but the walls are considered to be flexible.*

discussion of network architecture optimization can be found in Section 5.5.

The training procedure was divided in 3 stages with 150 epochs each. In the first stage, the network was trained by presenting it with the previous time instants generated by the 1DAS software. At the beginning of the second stage, the semi-trained network was called to carry out the *Prediction Procedure*. Each input distribution, which previously was generated by the 1DAS software, was replaced by a network prediction. Ergo, the network was enforced to learn to predict a following distribution by using its own predictions (slightly inaccurate distributions) and not the accurate distributions, generated by the 1DAS software. Then, the network continued its training with the updated dataset. At the beginning of the third stage, *Prediction Procedure* and dataset update were repeated, as mentioned earlier, but with a more trained network this time. The 3 distinct training stages altered the typical training process due to the adoption of the *Prediction Procedure* during the training phase. Note that the network lacked raw information about the varying A_0 in each artery (i.e. coordinates of A_0 or the values of inlet and outlet radii could have been used as input) and was called to map the correlation only between the previous time instant distributions and the following one. Each of the training stages took 22.5 *minutes*, on a single NVIDIA 1050 GPU, resulting in 67.5 *minutes*. Combined with the cost of the predictions during the training the total cost summed

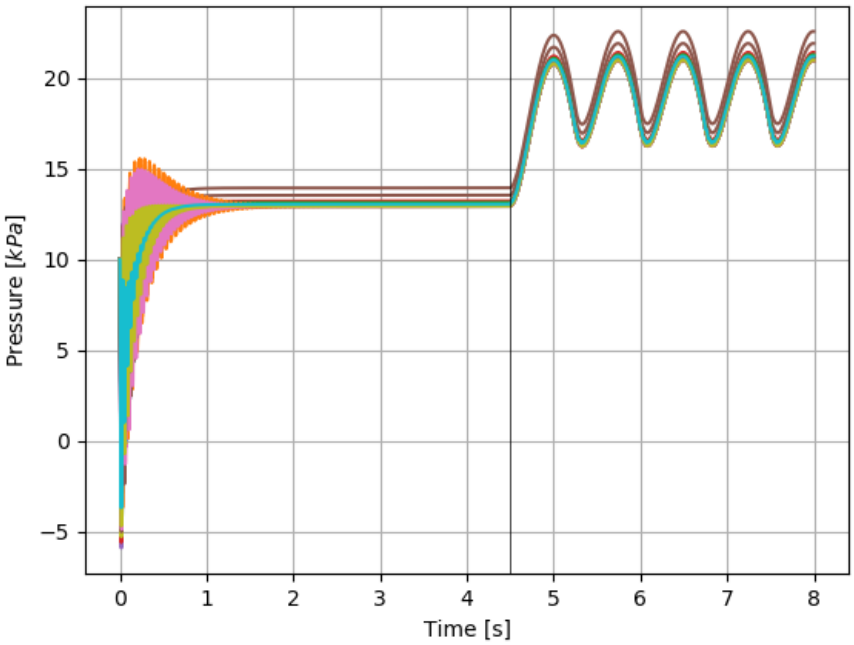


Figure 5.7: *Pressure time-series at the probe (50th) node. For the first* 4.5*s (black vertical line corresponds to the beginning of the second phase)* $\dot{Q}_{inlet}$ *is constant.*

to 1.6 *hours*.

5.2.5 Results

In figs. 5.13 and 5.14, results of 2 not seen by the network arteries, at the same 50th probe node, are shown. The average MAE of the prediction is $6 \times 10^{-3} m/s$. The results are more than satisfactory considering that the LSTM was presented with 150 instantaneous distributions from the first phase (375 instantaneous distributions in each period) and it was able to reconstruct the requested time-series consisting of 1875 distributions. Since the 150 initial distributions were part of the first phase (converged solution/exactly the same distributions), the variation of their number has no impact on the cost of the *Prediction Procedure* (regarding the use of the 1DAS software). This means that the value of n_{steps} influences only the training cost.

5.2.6 Rigid Walls

The capabilities of the LSTM network were evaluated on the 1DASR software, that models the quasi-1D artery with rigid walls. The cross-sectional area (A) was pa-

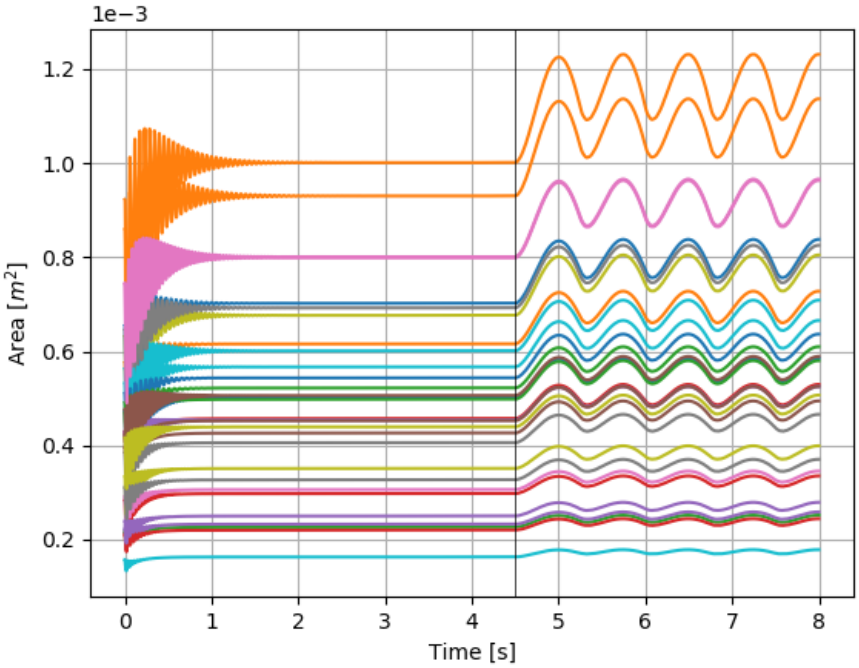


Figure 5.8: *Area time-series at the probe (50th) node. For the first* 4.5s *(black vertical line corresponds to the beginning of the second phase)* $\dot{Q}_{inlet}$*is constant. The converged area at the end of the first phase is slightly increased from the initial one at the beginning of same phase. The flexible walls adapt themselves to the evolving pressure within the artery.*

rameterized and was used as the initial cross-sectional area, in the previous case, and the blood inflow was computed form eq. (5.1) for the whole simulation time. The total simulation time was 8s and 800 time-steps were used. The artery length was discretized with 101 equidistant nodes. The physiological artery data of Table 5.1 were used by the 1DASR.

28 arteries were generated by varying the CPs of the Bezier curve that parameterized the area, as shown in fig. 5.2. 1DASR solved the equations and computed the 28×800 time-varying pressure distributions, fig. 5.15. There was an initial transient phase and, then, the pressure became periodic, following the blood inflow. The goal of the LSTM was to predict the following pressure distribution by presenting it with 40 (n_{steps}) preceding instantaneous distributions of the same quantity. The architecture and the training remained unchanged. Apparently, the input of the network was adjusted to the new n_{steps}. The training took 35$mins$ on the NVIDIA 1050 GPU.

Results, fig. 5.16, on a, not seen by the network, artery showcase that the same LSTM can be trained to predict the pressure distribution of an artery with rigid walls. The MAE of the predicted time-series is $1.6 \times 10^{-3} m/s$.

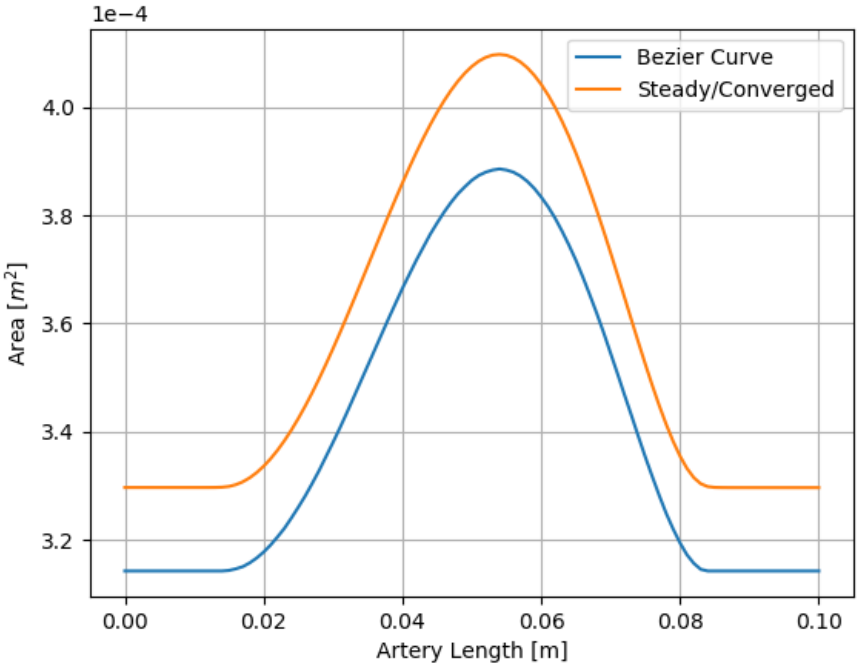


Figure 5.9: *The converged (end of first phase) area distribution of A_0 (orange) parameterized with the reference Bezier curve (blue), as shown in fig. 5.1. The artery is outward bulged.*

5.3 Realistic Blood Inflow

5.3.1 Digitization and Parameterization

In this case, a physiological Q_{inlet} was used as input for the 1DAS software. The Q_{inlet}, that was previously computed from eq. (5.1), was replaced by a waveform presented in [44]. The period of the waveform was equal to $T_h = 0.75s$, as well.

In order to obtain this waveform, it was necessary to use a digitization software. The data points were digitized and the curve emerged. Since the time intervals between the values of the curve were not equal, it was required to interpolate the curve. Thus, the curve was parameterized using a Bezier curve with 10 CPs and the Q_{inlet} values in each time instant were obtained. The Bezier curve was fitted manually, via the trial and error method, by moving the CPs, until the curve matched the digitized one. The curve, fig. 5.17, was later used as input for the 1DAS software run.

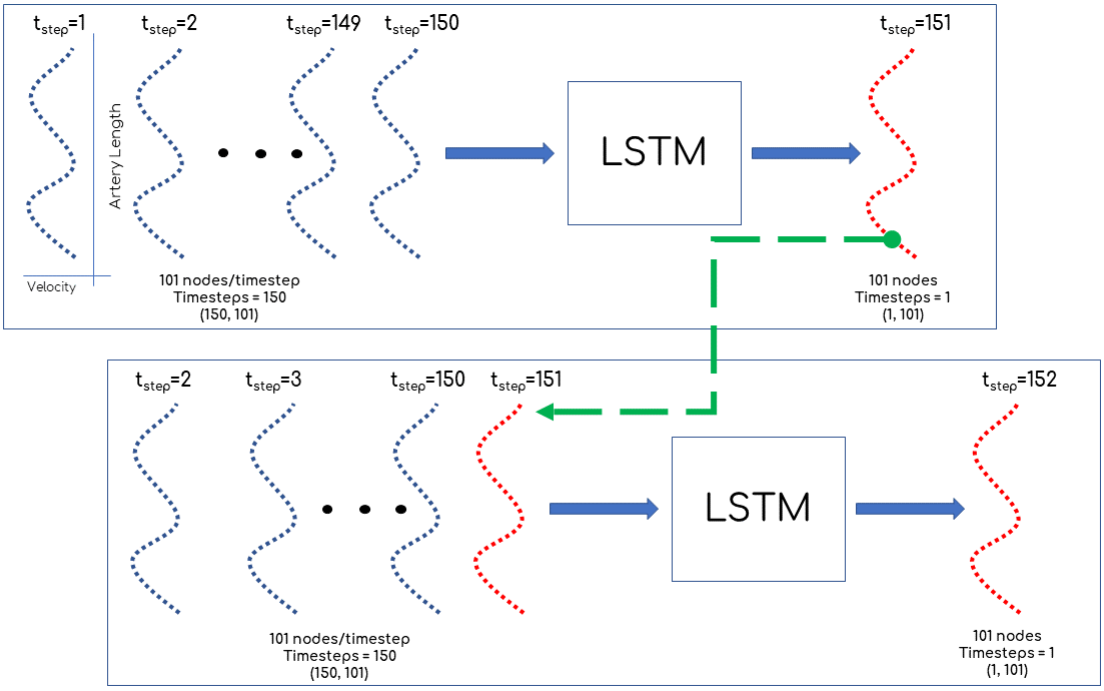


Figure 5.10: *Initialization of the Prediction Procedure. The network used as input distributions generated by the 1DAS software (blue) and then used its prediction (red) as input.*

5.3.2 Network Training and Results

The training and test datasets were generated, fig. 5.18, as mentioned above. The generated waveform of Q_{inlet} was utilized in the second phase, as the unsteady inlet flowrate profile.

In order to utilize the NVIDIA CUDA Deep Neural Network library (cuDNN), [45], it was required to use the default LSTM layers configuration with the *tanh* activation function, [10]. The network training was split into 3 stages of 150 epochs, as mentioned before. The network architecture remained identical to the network shown in fig. 5.12 and 150 distributions of previous time instants were used by the network. The utilization of the GPU-accelerated library, cuDNN, resulted in the decrease of the total training cost to 1 *hour* (compared to 1.6 *hours* without cuDNN).

The resulted network predictions, fig. 5.19, at the probe (50th) node, improved, compared to the previous case. The average MAE of the predicted time-series is $1.4 \times 10^{-3} m/s$. The predictions of the LSTM network are accurate and the network was able to reconstruct the velocity time-series. As long as the blood inflow is computed from a periodic function, the network performance is independent of the equation from which it is computed. Besides that, the network architecture and

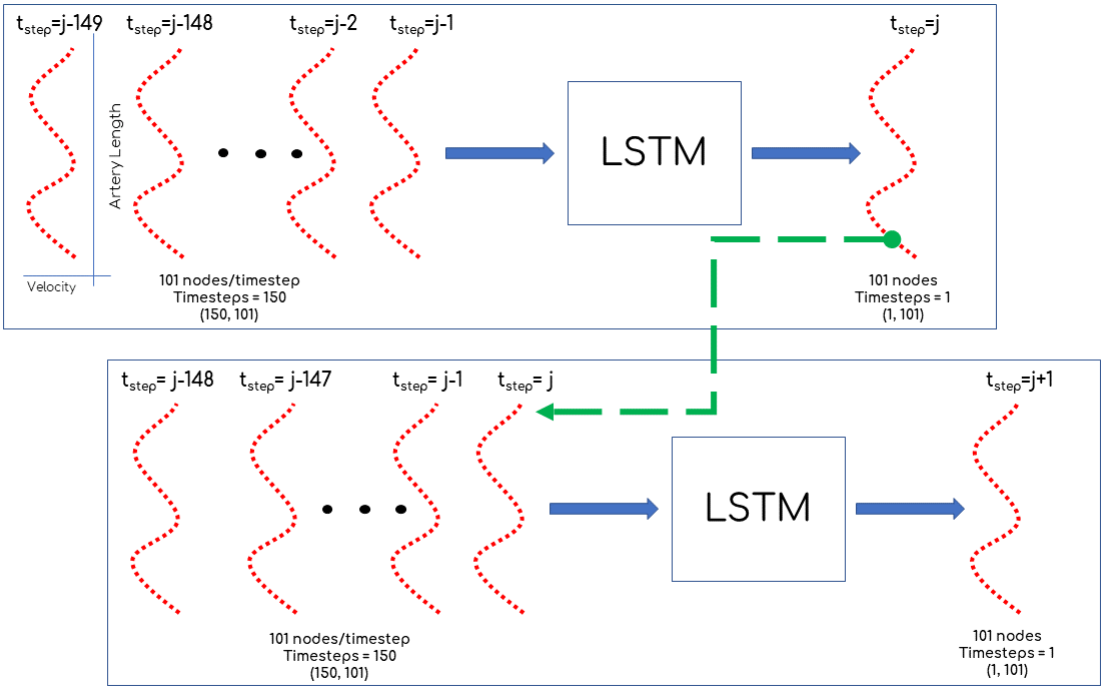


Figure 5.11: *Prediction Procedure presented at time step $t_{step} = j$. The network used its own prediction at $t_{step} = j$ as input for prediction at $t_{step} = j + 1$.*

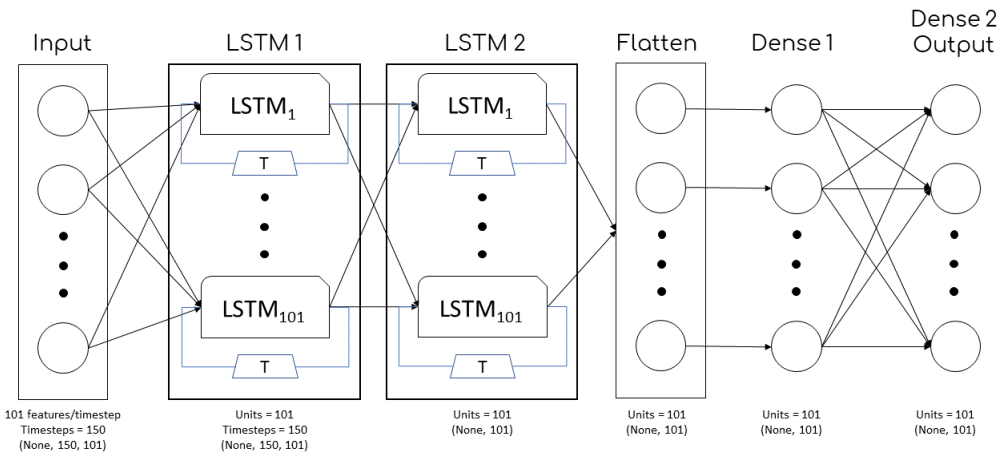


Figure 5.12: *The LSTM network architecture. The network consists of the LSTM part, a flatten layer and the dense part. The number of artery nodes is called features of the network. The network was presented with the 150 preceding instantaneous flow solutions.*

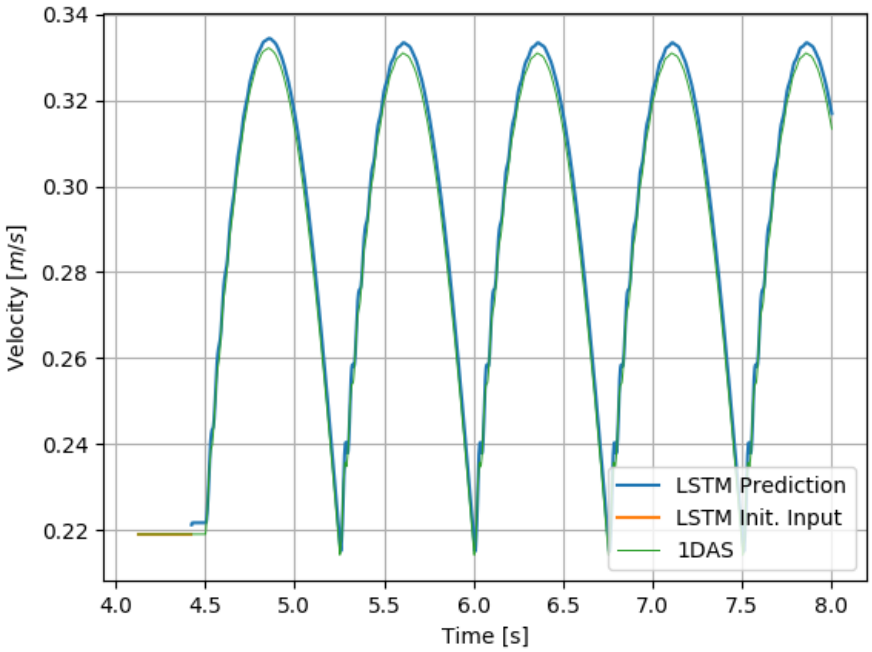


Figure 5.13: *The 150 steady state distributions as initial input to the LSTM network (orange), the prediction of the network (blue), and the time-series generated by the 1DAS software (green/thin). All curves are plotted at the probe (50th) node.*

the training procedure remained unchanged in this case. Consequently, the increase in the prediction accuracy is related to the implementation of the *tanh* activation function.

5.4 Varying Artery Wall Thickness

In this case, a weakening spot on the artery wall was introduced by varying the thickness of the wall along the length of the artery. Hence, besides the changes made in the previous case, the wall thickness distribution was parameterized and varied.

5.4.1 Parameterization and Training Dataset Creation

The wall thickness distribution curve was arbitrary selected and was parameterized using a Bezier curve with 7 CPs, as well. The parameterization procedure of the distribution was similar to the parameterization of the starting artery shape. A constant curve of thickness equal to $h_s = 1mm$ was utilized as an initialization for

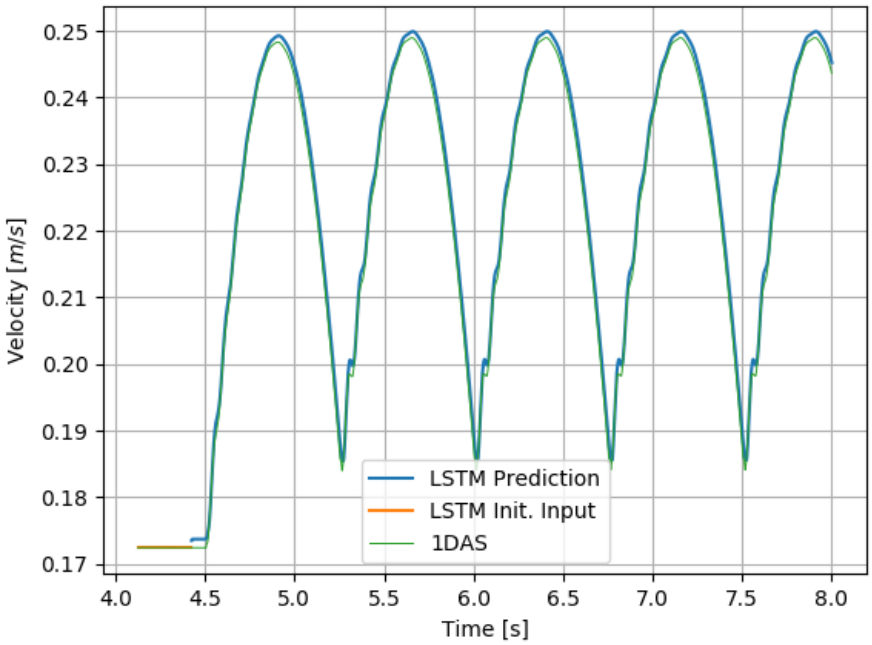


Figure 5.14: *The results for a different starting artery shape (A_0). All curves are plotted at the probe (50th) node.*

the procedure. The first and last CPs were fixed at 15% and 85% of the artery length, while the rest of them were evenly distributed between them. The thickness of the wall at the inlet was equal to the one at the outlet and, thus, the first and the last CPs had the same ordinate. The rest of the nodes had the same ordinate as the first and last CPs. By varying h_s by $\pm 20\%$, new CPs emerged. In order to simulate a bulge, the ordinate of the intermediate CP was multiplied by the factor 0.5, which varied up to -20%.

By varying the CPs of the Bezier curves that parameterized the thickness distribution and the starting shape of the artery, a dataset was generated, fig. 5.20. It consisted of 18 arteries for the training and 2 arteries for the evaluation of the network.

The set of the velocity profiles made up the dataset used by the network during the training procedure, 5.21.

A run with the CPs of initial cross-sectional area and thickness distributions in their reference position was performed, as shown in fig. 3.4. In order to showcase the results of the 1DAS software, 3D plots were created, fig. 5.22. The initial cross-sectional area can be seen. The elastic walls of the vessel adapted themselves to the evolving pressure and the constant bloodflow, in the first phase. The artery inflated and deflated, in the second phase, following the pulsatile action of the heart,

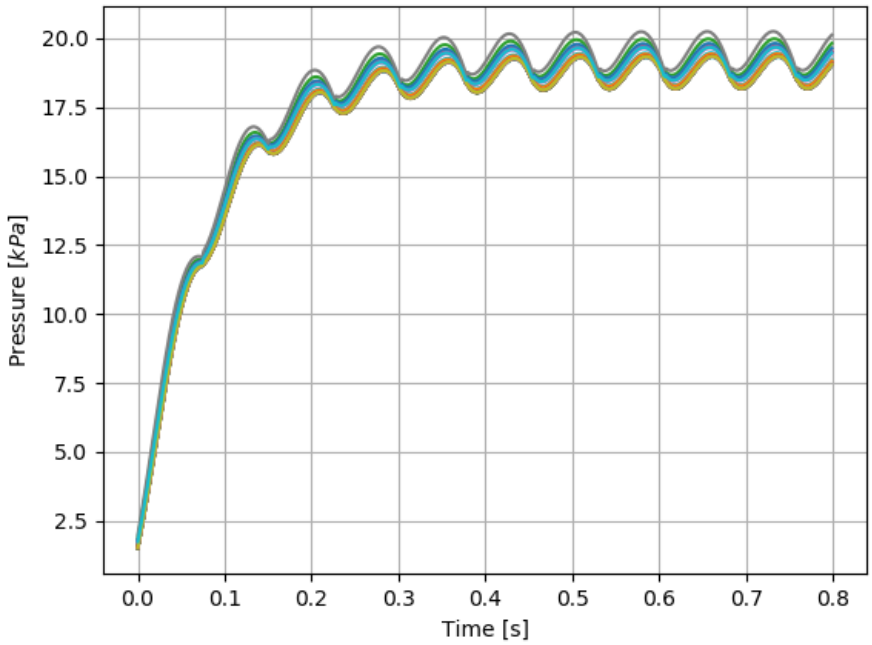


Figure 5.15: *The pressure time-series in the probe (50th) node.*

which was provided by the blood flow time-varying form. The velocity distributions followed the aforementioned pattern. In the first phase, the transition area and the steady state area can be highlighted. In the second phase, velocity monitored the pulsatile action, as well. The waves propagated through the artery.

5.4.2 Network Training and Results

In order to provide the network with more information, the thickness distributions and the starting artery shapes were utilized as network inputs. The network was presented with the 150 velocity distributions of previous time instants in conjunction with the starting shape and the wall thickness distribution of each artery. In total, 152 distributions were fed as input. The input layer of the network was modified to adapt to the new input dimension of 152 distributions. The training procedure, as well as the training cost, were identical to the previous case. Results, fig. 5.23, confirm that the network is capable of reconstructing, accurately, the velocity time-series. The average MAE of the predicted time-series is $1.8 \times 10^{-3} m/s$.

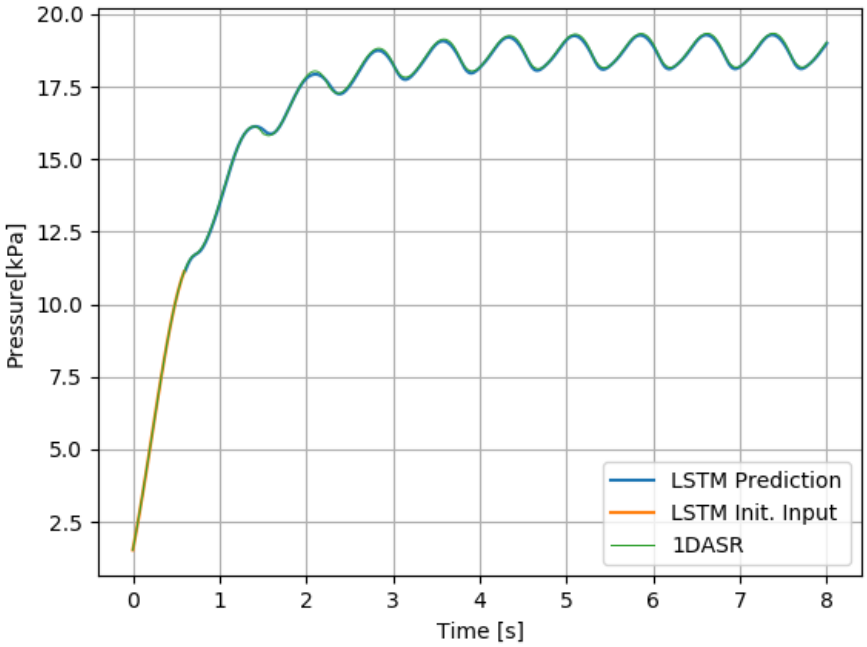


Figure 5.16: *The predicted pressure time-series of a, not seen by the network, artery in the probe (50th) node.*

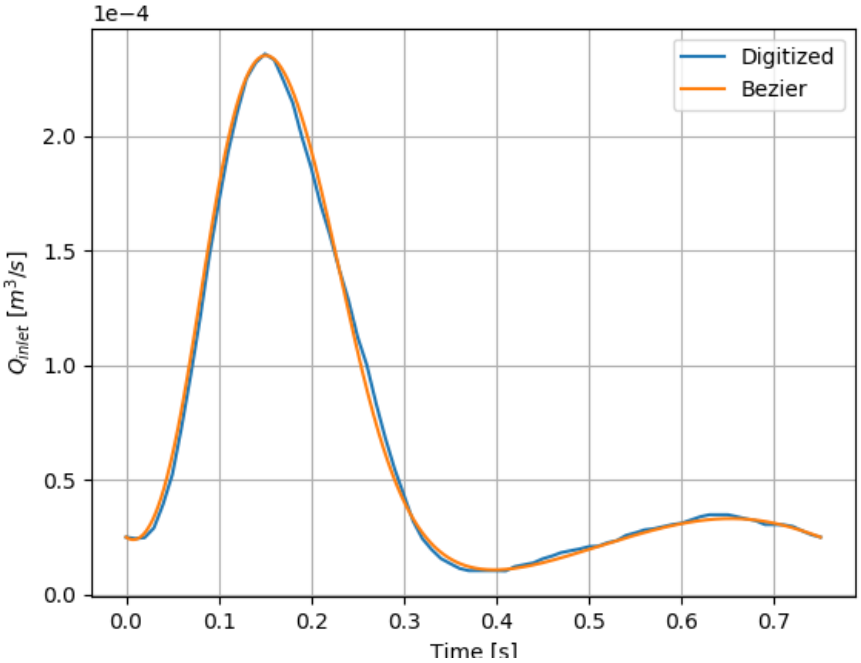


Figure 5.17: *The digitized waveform of Q_{inlet} (blue) and the Bezier Curve (orange).*

5.4.3 Alternative Training and Results

In order to test the capabilities of the network, it was trained, with the same dataset, to predict the velocity distributions without being presented with either the starting

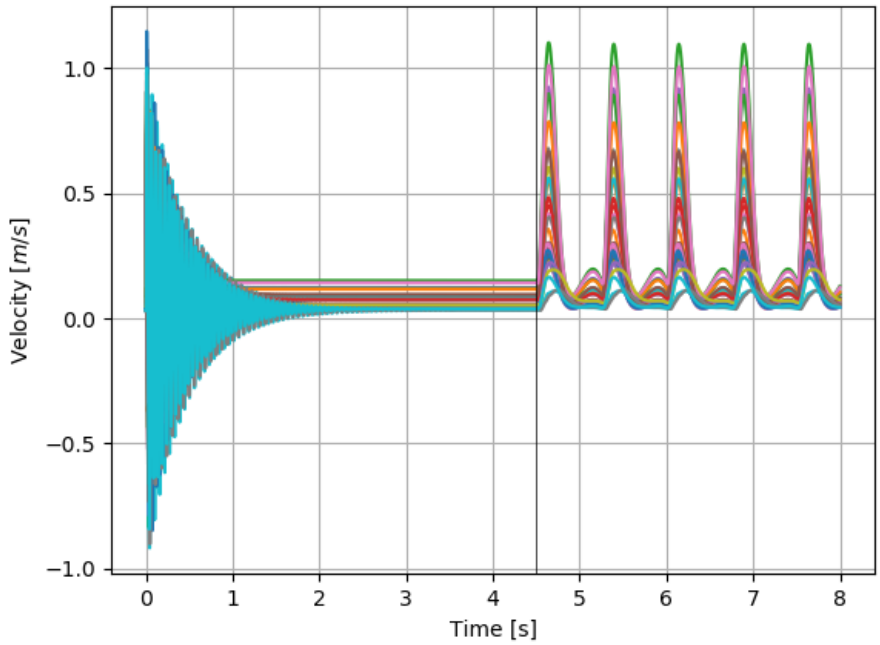


Figure 5.18: *Velocity time-series at the probe (50th) node. The physiological Q_{inlet} was used as the unsteady inlet flowrate profile after the first 4.5s (black vertical line corresponds to the beginning of the second phase).*

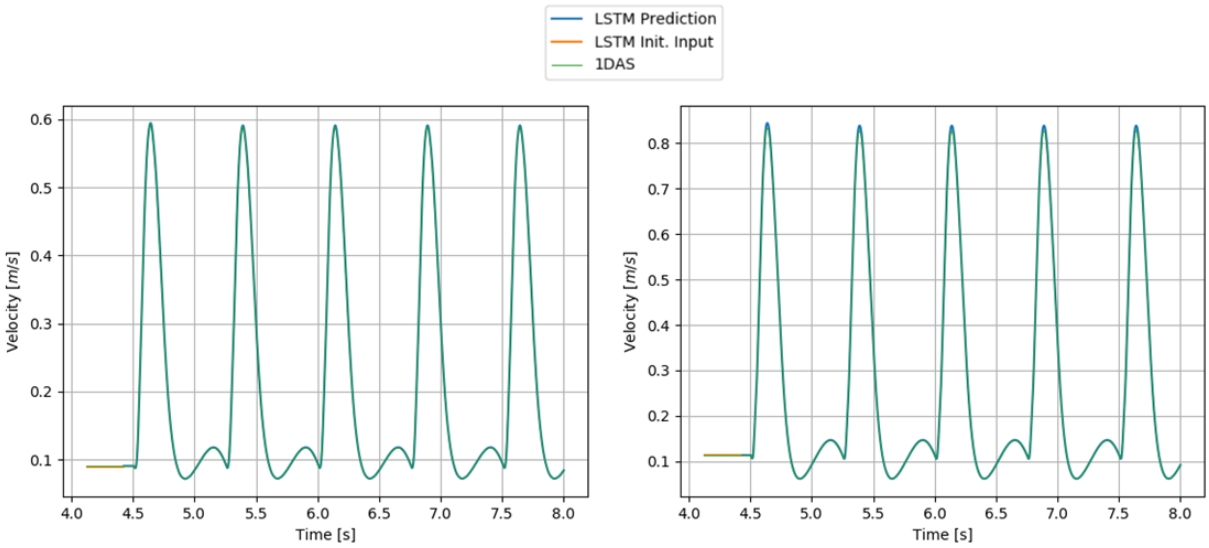


Figure 5.19: *The predictions, of two not seen by the network, artery shapes. All curves are plotted at the probe (50th) node.*

shape of the artery or the thickness distribution. The results were very accurate, fig. 5.24, since it was able to reconstruct the velocity time-series only by presenting

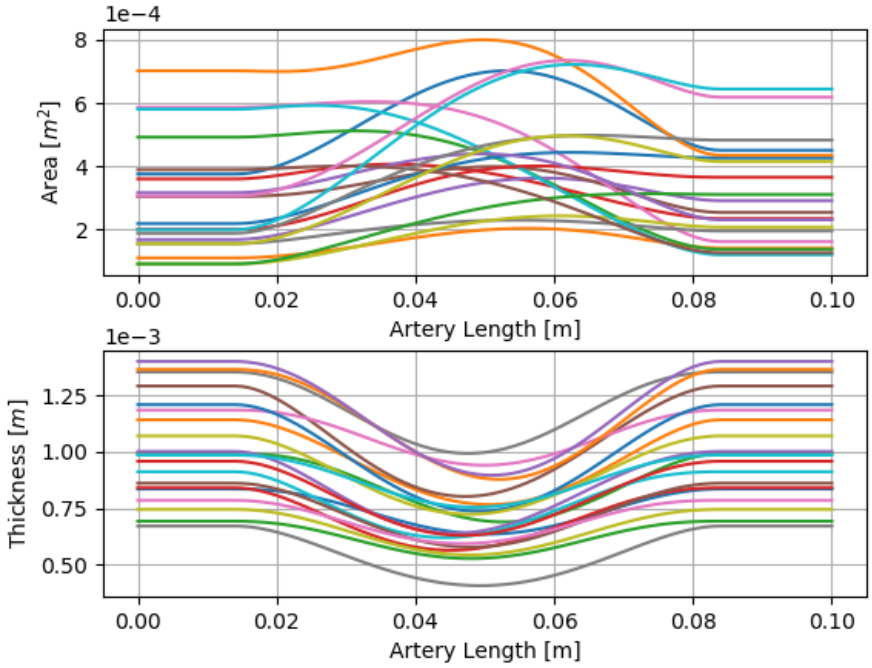


Figure 5.20: *The 20 starting shapes of the arteries and the corresponding thickness distributions.*

it with the 150 initial distributions. The average MAE of the LSTM predictions is $1.2 \times 10^{-3} m/s$. These 150 distributions contained the necessary information, and the network was able to map the correlation between them and the 2, not presented to it, distributions. Note that, the initial cross-sectional area of the artery and the thickness distribution along it, in terms of A_0 and h, appeared in eqs. (3.5) and (3.6).

5.5 Optimization of LSTM Input and Architecture

In order to decrease the network training cost, it was required to reduce the size of the matrices, used during the training. One dimension of the training matrix is the number of distributions that the network was presented with, n_{steps}. This number encodes the information of the number of preceding distributions needed for the prediction of the following distribution, by the LSTM. In an effort to reduce this number, it was necessary to search the dependency of the current prediction to the number of preceding ones. Thus, a statistical method, called Auto-Correlation (AC), was utilized to find the optimal n_{steps}.

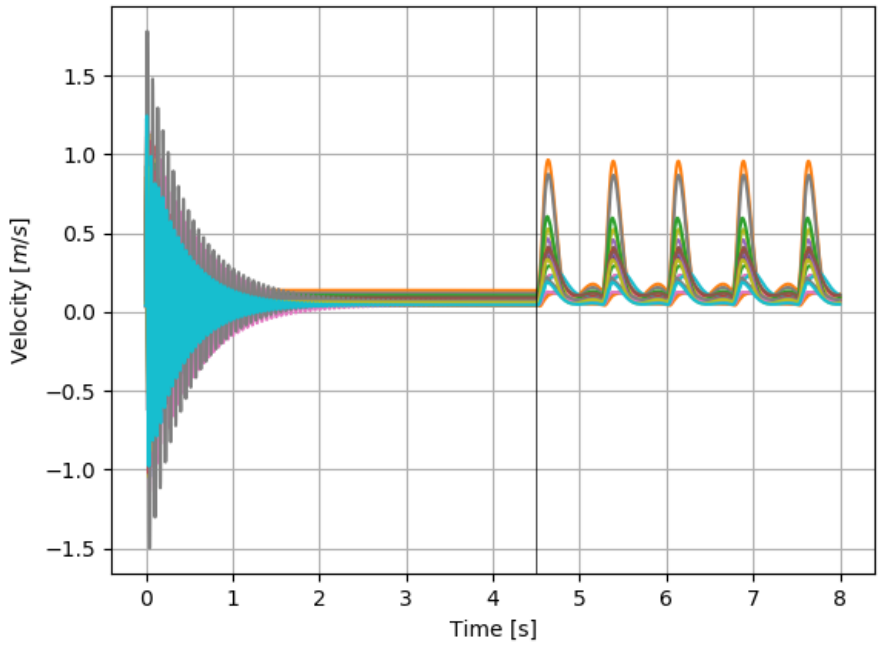


Figure 5.21: *Velocity time-series at the probe (50th) node. The thickness of the wall varied along the artery.*

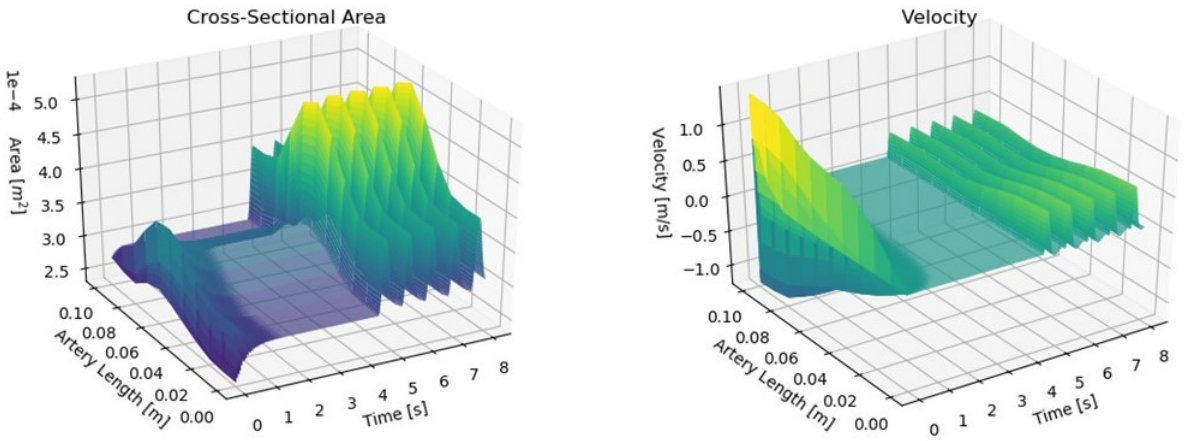


Figure 5.22: *The cross-sectional area and velocity distributions generated by the 1DAS software.*

AS correlates a time-series to a delayed copy of itself, in different time intervals (time-lags), by computing the Pearson correlation coefficient, [46]. As the AS coefficient of a specific time-lag is increased, the more the current value is influenced by the values in these preceding time intervals. For instance, if the temperature is high today, it is more likely to be high tomorrow than to be high in one month. The AS coefficient

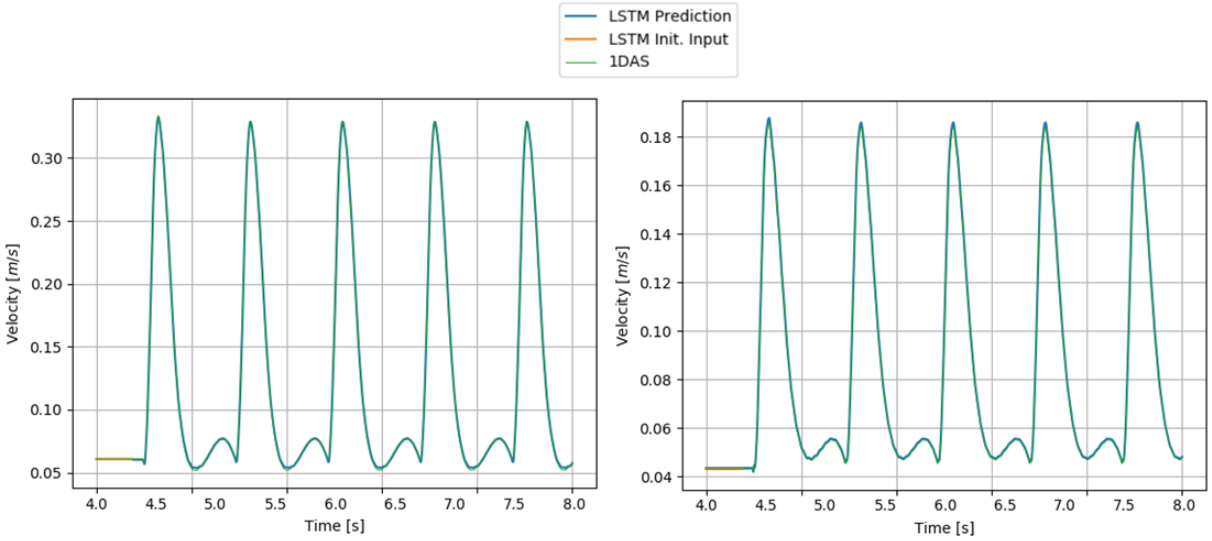


Figure 5.23: *The predictions of two, not seen by the network, artery shapes and wall thickness distributions. The network is presented with the starting artery shape and the wall thickness distribution. All curves are plotted at the probe (50th) node.*

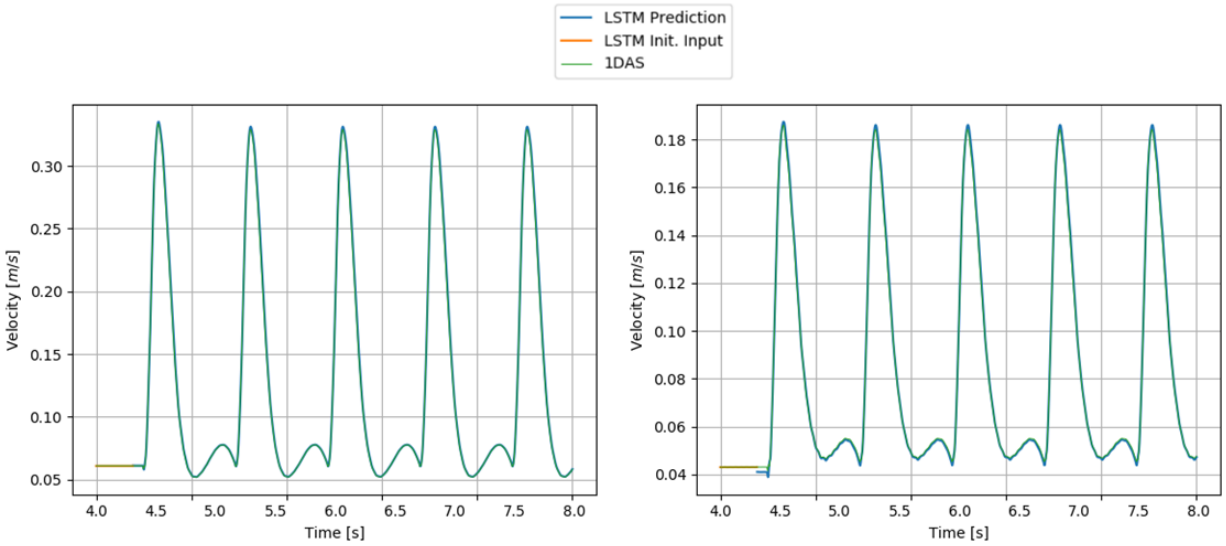


Figure 5.24: *The predictions of two, not seen by the network, artery shapes and wall thickness distributions. The network was presented with only 150 initial distributions. All curves are plotted at the probe (50th) node.*

in 1 time-lag (1 day) is higher than that in 30 time-lags (1 month). Account, also, for the fact that the temperature of today is influenced by the temperature of the same day one year ago (365 day - 365 time-lags). The AS computes the correlation between not only adjacent and near-adjacent values but all delayed values.

Following the example, the temperature of today is mostly affected by the temperatures of yesterday and the day before yesterday. Nevertheless, the temperature

of yesterday is also influenced by the temperature on the day before yesterday. AS takes account the influence coming from the day before yesterday directly or through yesterday. The removal of this indirect impact is performed with another statistical method, the Partial Auto-Correlation (PAC). PAC correlates a time-series to a delayed copy of itself but the values of the time series are regressed in all shorter time-intervals.

Both AC and PAC methods were programmed in a PYTHON code. The velocity time-series in the (50th) probe node was used for presenting the results, fig. 5.25. The peaks of AC factor in multiples of period time steps (375) indicate the periodic form of the time-series. The PAC plot shows statistical significance for ~ 60 lags and, thus, n_{steps} set equal to 60.

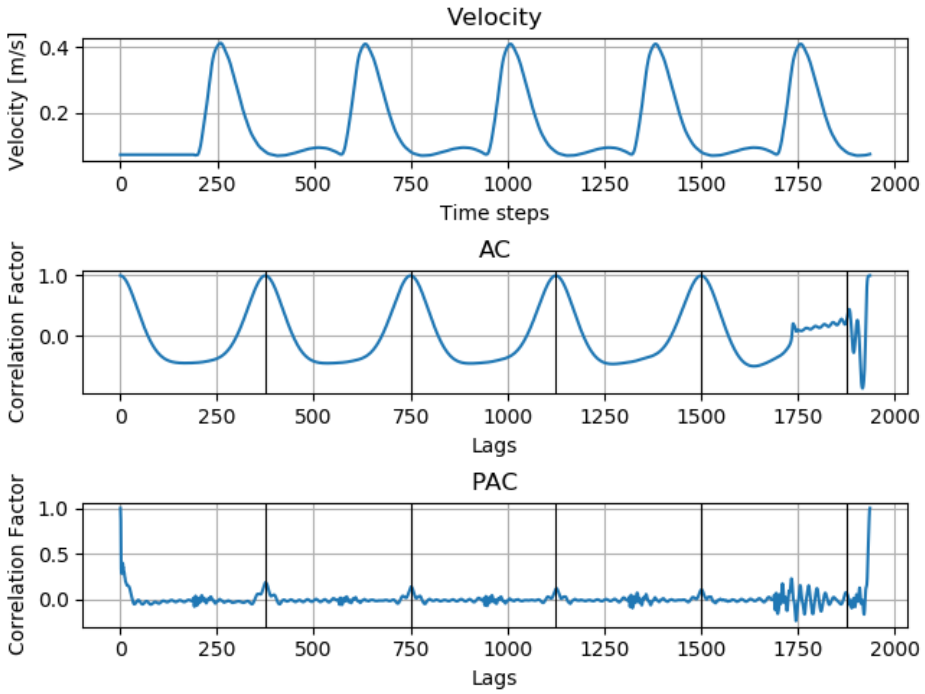


Figure 5.25: *AC (middle) and PAC (bottom) in the velocity time-series (top). Vertical black line correspond to periods.*

Subsequently, SHERPA, [47], was used for optimizing the LSTM architecture. It is a PYTHON library for optimizing hyperparameters of ML models. The Population Based Training, as introduced in [48], was utilized for optimizing the number of neurons in each layer. The population size and the maximum number of generations were set to 5 and 30, respectively. The perturbation factors were 0.8 and 1.2, as well. The range of the design variables of the optimization algorithm, number of neurons, was between 50 and 250. The optimization objective was to minimize the MAE between the predicted distributions and the distributions generated by the 1DAS. During the optimization, the network was trained for, only, 100 epochs. The $n_{steps}$was equal to 60 and 18 arteries were used. The optimization took ~ 13.5 *hours*

on the NVIDIA 1050 GPU. The optimal numbers of neurons for the first and the second LSTM layers were 108 and 128, respectively. 196 neurons for the dense layer were the optimal number, according to SHERPA.

The network, with the optimal number of neurons and the n_{steps}, determined by the PAC, was retrained with the dataset of the previous case study. The training of the network became less time-consuming and memory-demanding, since the input matrix size decreased. The MAE of the optimized LSTM prediction is $1 \times 10^{-3} m/s$, fig. 5.26. The training cost decreased to 36 *mins* and the results were very satisfying, as well.

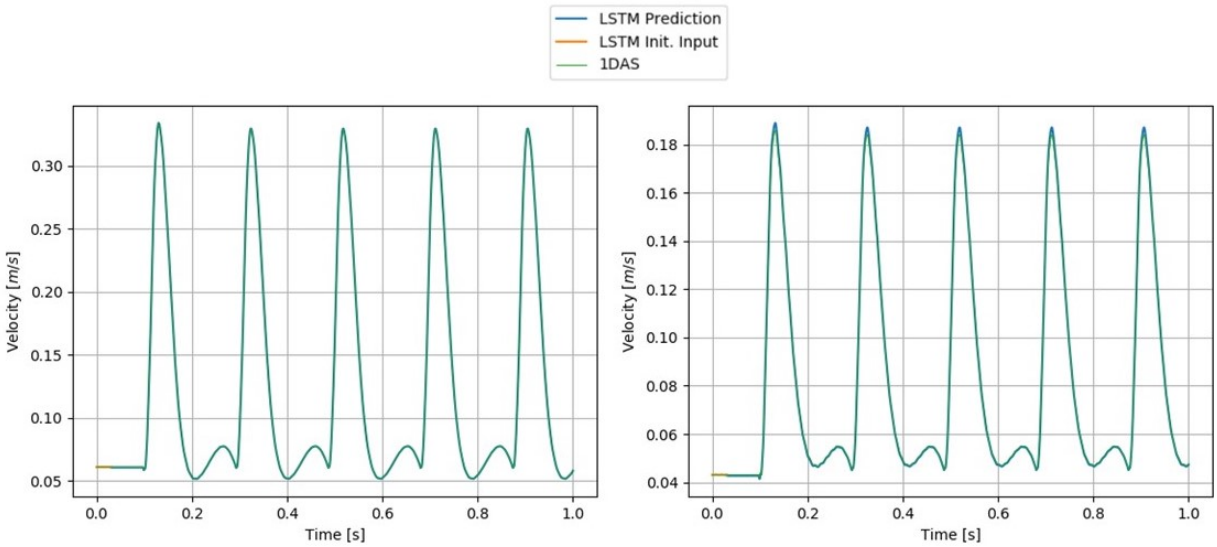


Figure 5.26: *The prediction of two, not seen by the optimized network, velocity time-series at the probe (50th) node. The network input of 60 distributions.*

Chapter 6

Prediction of Scalar Fields with λ-DNNs

Following the previous applications, another type of DNN, the λ-DNN was utilized for predicting flows. Its name comes from the Greek letter λ. The λ-DNN was used for predicting aerodynamic flows in two cases, by presenting it with the nodal coordinates and the case related data. The goal of the network was to reconstruct entire flow fields by making predictions node by node. In the last case, the network was used in a multi-disciplinary analysis, by replicating one of the disciplines. The network was called to predict entire temperature distributions along the contour of an internally cooled blade.

6.1 Network Architecture

The λ-DNN was proposed and presented in [49] and [50], by the PCOpt/NTUA group (including the author). It utilized the architecture of multi-branch DNNs, [51] and [52]. This architecture, fig. 6.1, was based exclusively on fully-connected layers and comprises branches for each type of input. The number of layers per branch usually varied in each case, depending on the type of input, and it was selected after some trial and error and literature review. The output of the branches was fed (concatenation) to another network, the core network, whose output was also considered to be the output of the total DNN. The number of layers and neurons of the core network was selected as in the branches. Two types of input were usually presented to the network, the nodal coordinates of the body shape to be designed and the case related data.

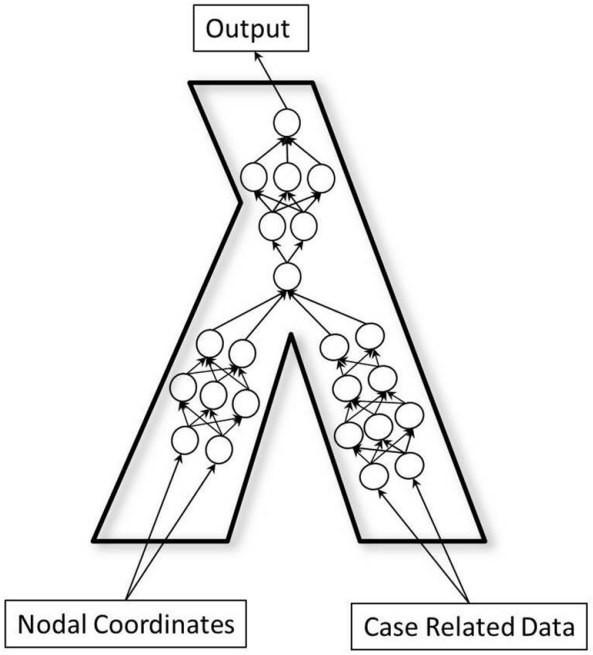


Figure 6.1: *The λ-DNN architecture. The name comes from the Greek letter λ, due to its multi-branch shape. From [50]*

The advantages of the multi-branch network structure were that, due to the smaller number of synaptic weights (compared to a fully connected network), it was less prone to over-fitting, while at the same time achieving more accurate results. In addition, it had the ability to predict either nodal quantities or whole fields. Its structure was superior to a corresponding convolutional network, [53] and [54], in the prediction of unstructured grid fields. Unlike the implementations of the CNNs, the λ-DNN inputs were independent to the connectivity between the nodes. The multi-branch architectures allowed convergence to global solutions, during the network training, due to the fact that they are less non-convex (less local minima), [55].

6.2 Applications

6.2.1 Prediction of Flow Around an Isolated Airfoil

The goal of the λ-DNN was to reconstruct the 2-D pressure distribution around an isolated airfoil. The network was called to predict the nodal pressure by presenting it with the coordinates of the corresponding node and the coordinates of the airfoil contour.

Network Architecture

The network of this case consisted of two branches for the two different inputs, 6.2. The branch of the nodal coordinates consisted of 4 layers with 128, 256, 256 and 128 neurons respectively. The branch with the case related data as input had 3 layers with 128, 256 and 128 neurons. The core network had 3 layers with 128, 256 and 128 neurons, as well. The ReLU activation function was used in all but the last layer which used the sigmoid function.

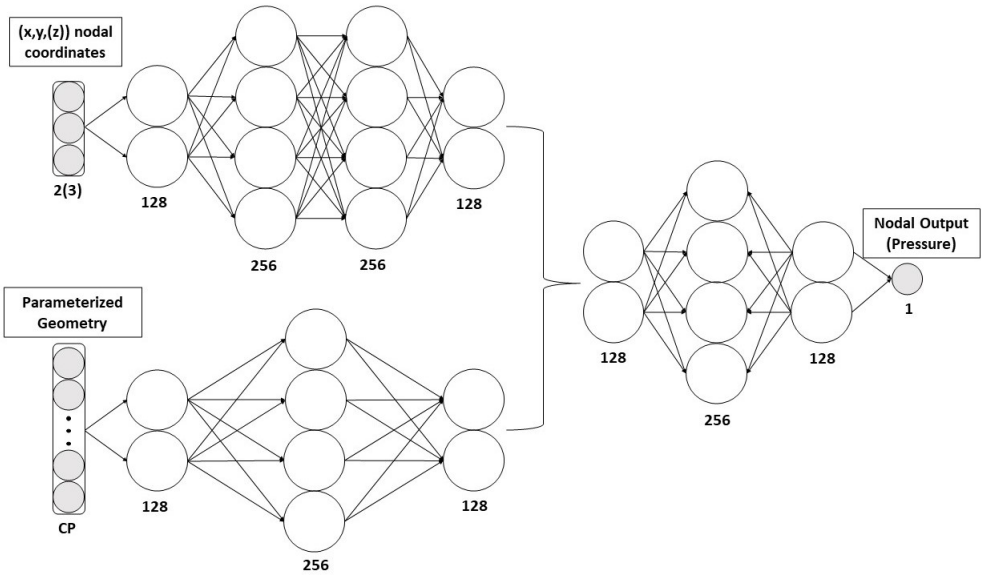


Figure 6.2: *The λ-DNN architecture. It consists of two branches, for each type of input. The number below each layer indicates the number of neurons. The number of nodal coordinates and the parameterization control points vary between the cases.*

In order to evaluate the λ-DNN architecture capabilities, it was compared with a Fully-Connected Network (FCNN). For a fair comparison, the architecture of FCNN was selected with the trial and error method, as well. The networks compared with their best possible results and not in the manner of the architecture similarity (same number of trainable parameters). The architecture comprised 4 fully-connected layers with 512, 312, 256 and 56 neurons. Input to the FCNN were both the airfoil nodes and the coordinates. Its activation function was the same as the λ-DNN.

Training Dataset Creation and Network Training

In order to create the training dataset, the airfoil shape was parameterized using two Bezier curves, with 6 CPs each, for the pressure and the suction side. The first and the last CPs were constant (leading and trailing edges). Their ordinates varied by ±20% and 280 airfoil shapes were generated. Only 180 shapes were utilized for the network training while the rest were used as an evaluation dataset, fig. 6.3. For each airfoil shape, different unstructured grids with $\sim 10K$ nodes were generated.

The software PUMA, [56], was used for solving the Reynolds-Averages Navier-Stokes equations. The flow was inviscid with free-stream Mach number $M_\infty = 0.62$ and flow angle $a_\infty = 2.7°$. It took $\sim 10sec$ on a K20 GPU to solve the flow around the airfoil.

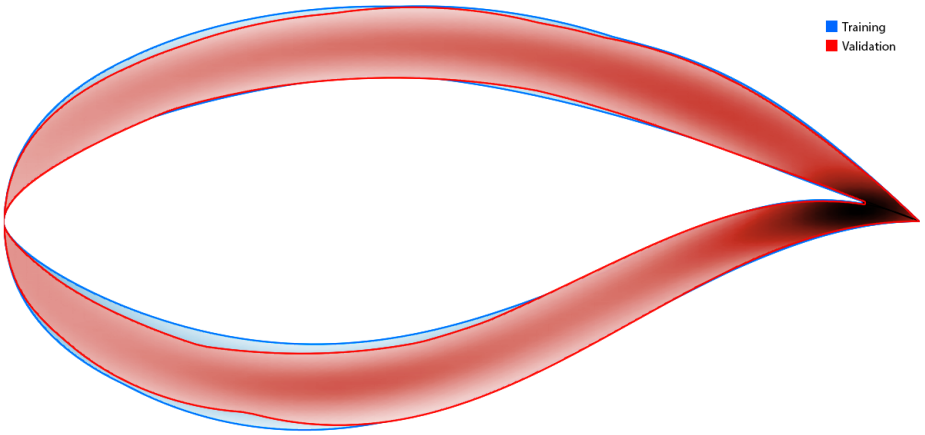


Figure 6.3: *The airfoils used for the training (blue) and the validation of the network (red). The boundaries denote the outer airfoils of each dataset.*

Since the λ-DNN predicted the pressure at each node, the dataset consisted of the nodal pressure values of each airfoil shape. The number of training patterns was the product of the nodes ($\sim 10K$) and the number of the airfoil shapes or flow fields generated by PUMA software (180). The network training for each network (λ-DNN and FCNN) took $2.5hours$ on a single K20 GPU.

Results

The network was evaluated using the 100 flow fields not seen by the network during its training. It was called out to predict the pressure value at each node given the corresponding coordinates and the contour coordinates. The results, fig. 6.4, were compared to the PUMA software results and the FCNN predictions. The λ-DNN was able to reconstruct accurately the pressure field around the arfoil. The MAPE of the λ-DNN results was 0.36% compared to the FCNN's that was equal to 0.50%. The multi-branch architecture of the proposed network outperformed the classic fully-connected one.

6.2.2 Prediction of Pressure Distribution in a Francis Turbine Runner

Inhere, the λ-DNN with the aforementioned architecture adjusted to this case, was trained to predict the pressure distribution on the surface of an inlet guide vane of a

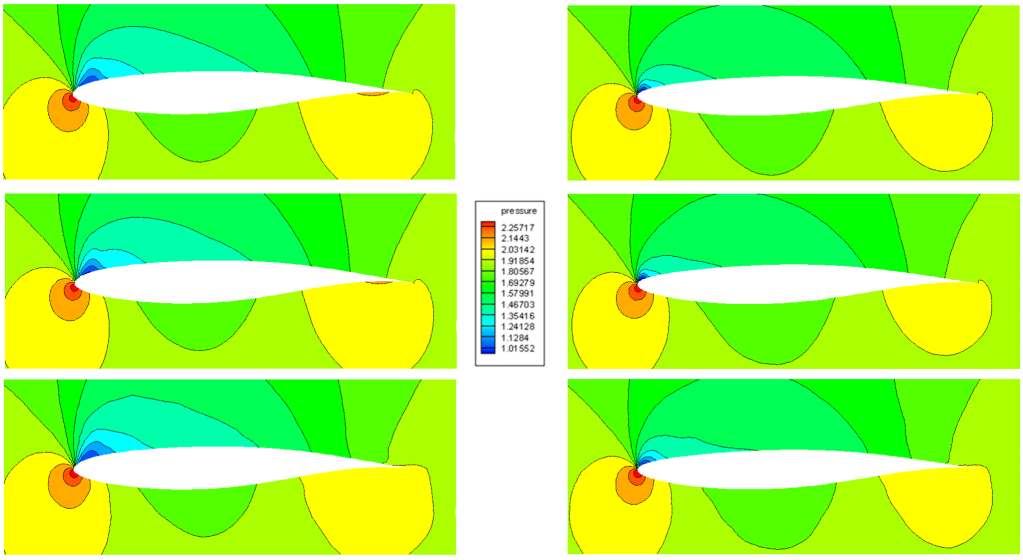


Figure 6.4: *Pressure fields constructed by the λ-DNN (middle) and FCNN (bottom) are compared to those generated by the PUMA software (top). Two airfoils were not seen by the network during its training.*

Francis runner, fig. 6.5. The network input consisted of the nodal coordinates and the design variables that parameterize the runner.

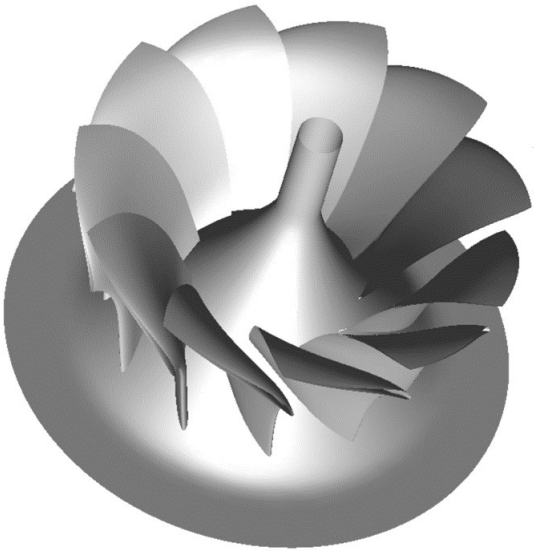


Figure 6.5: *The runner of the Francis water turbine.*

Training Dataset Creation and Network Training

The flow was turbulent with inlet total pressure $P_{t,inlet} = 261Pa$, inlet angles $a = 20.2^o$ and $b = 90^o$, outlet static pressure $P_{outlet} = 92Pa$ and rotation speed $1652RPM$. The

runner geometry was parameterized using the GMTurbo software, [57]. In order to create the training patterns, 16 design variables varied by ±10%. These variables correspond to the span-wise distributions of quantities parameterizing the camber surface. For every runner an unstructured grid was generated with $\sim 1.2M$ nodes, while the number of the surface nodes was $\sim 10K$. The solution of each field took $\sim 40min$ on a K40 GPU. In total, 31 flow fields were generated by the PUMA software; 30 out of them were used by the network during its training. Since the runner was 3-D, the input consists of the 3 nodal coordinates and the 16 design variables. The input layer was adjusted to be presented with the input matrices, fig. 6.2. The network training took $\sim 40min$ on a single K20 GPU.

Results

The network predictions were compared to the surface pressure distributions generated by the PUMA software. One new, not seen by the network, runner constituted the evaluation dataset in order to test the network performance, fig. 6.6.

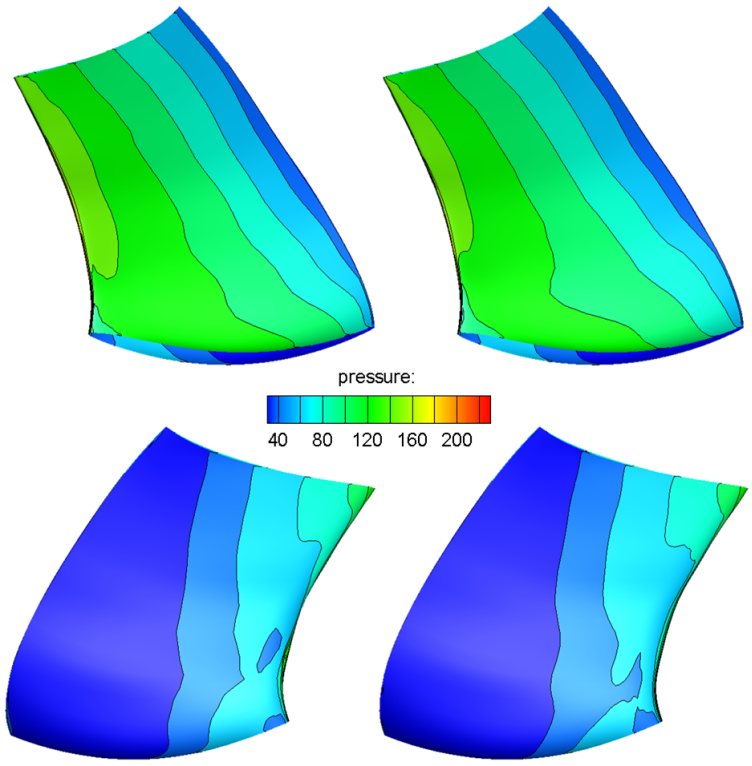


Figure 6.6: *Surface pressure distribution on a, not seen by the network, runner generated by CFD (left) and predicted by λ-DNN (right).*

The results, with MAPE equal to 3%, are more than satisfying, considering the number of runners used for the training. Note that the network architecture was the same, aforementioned, architecture and it was able to reconstruct the surface

pressure distribution accurately. The proposed network architecture was capable of outperforming a typical fully-connected network and,in fact, it can be utilized in other applications with an almost identical architecture.

6.2.3 Prediction of the Temperature Field in a CHT Problem

CHT analysis

Since gas turbine engines are designed to operate in high inlet temperatures, it is required to cool the parts, that are in contact with the hot gas, flowing through it. By incorporating cooling techniques, such as air passage or film cooling, in the blades, the thermal load is decreased, and their lifetime is expanded. In order to study and optimize the thermal behaviour of the structural elements, simulations are necessary, utilizing modern CFD software capabilities. For instance, in the thermal design of a 2-D turbine blade, a CFD solver, that computed the flow around the blade, coupled with a heat conduction solver, that solved the heat conduction equations over the solid blade, can be utilized. If these solvers/computations are decoupled, many iterations are required for an accurate solution. In the case that the codes are coupled, they solved the equations simultaneously, exchanging information over the adjacent boundary. The communication/interaction (Fluid-Structure Interaction - FSI) between the two solvers of the equations, during the iterative solving, increases the demand of resources (time and computational power). Inhere, a coupled multi-disciplinary simulation, known as Conjugate Heat Transfer(CHT), was performed.

The internally cooled C3X cascade was used for the CHT analysis, [58]. The turbine blade was cooled by 10 radial channels. The location of these channels with circular cross-sections was fixed, as well as their diameters. The solid domain of the blade was in contact with a flow domain around it, fig. 6.7. The blade is made out from stainless steal and its density was equal to $\rho = 7900 kg/m^3$ and the heat capacity was equal to $C = 586.15 J/kgK$. The thermal conductivity was a linear function of the blade temperature and it was defined as $k = 6.811 + 0.020716T$, [58].

In order to analyse a design in a CHT problem, it was necessary to solve the flow equations over the flow domain and the heat conduction equation, over the adjacent solid one. The fluid solver provided the computed heatflux distribution, over the interface of the domains (i.e. the blade contour), to the heat conduction solver. The heat conduction solver exchanged back the temperature distribution of the solid domain. The goal of the λ-DNN was to replicate the numerical solver of the heat conduction equation by predicting the temperature distribution by presenting it with the heatflux one and the coordinates of the blade contour.

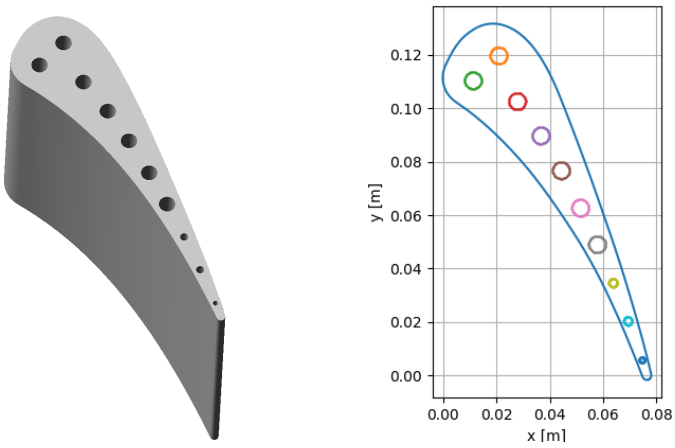


Figure 6.7: *The turbine blade with 10 cooling channels in 3D (left) and in plot (right). Note that the run was 2-D.*

Dataset Creation, Network Training and Architecture

Each discipline was resolved and the solvers provided boundary conditions, over their interface, to each other. For the fluid domain, the PUMA software solved the Reynolds-Averaged Navier-Stokes equations with the $k-\omega$ SST turbulence model, [59]. The hot gas flow had inlet total pressure $p_t^I = 243700Pa$, inlet total temperature $T_t^I = 800K$, and outlet static pressure $p^O = 142530Pa$. For the solid domain, PUMA software was utilized as the solver of the heat conduction equation, as well. The Neumann boundary conditions along the contour of each cooling channel were computed by the defined corresponding coolant temperatures and flowrates.

In order to generate training and validation patterns, the airfoil contour was parameterized with volumetric NURBS, fig. 6.9. A 7×3 control grid was created with 21 CPs. The CPs at the leading and trailing edges remained fixed while the rest 19 CPs varied, in both directions, ±5% of their reference position. Both domains were discretized using a grid of $\sim 312K$ nodes. $\sim 183K$ nodes were used for the flow domain while $\sim 129K$ were used for the solid domain, fig. 6.8. The solution of this problem took $\sim 15min$ on a V100 GPU. 162 geometries were created and used by the CHT solver.

Two types of inputs were presented to the network, the coordinates of the blade contour and the distribution of heatflux along the interacting boundary, between the fluid and the solid domain. The λ-DNN of fig. 6.2 and a modified version of it were used in this case. The modified version of λ-DNN was presented in [50] and its both branches consisted of 1 layer of 64 neurons while the core network had 1 layer of 128 neurons. Note that, in this case, the networks predicted the entire field, in comparison to the previous cases. For comparison, a FCNN network with 2 layers with 128 neurons was trained, as well. The input presented to the FCNN was a concatenated form of the one presented to the λ-DNN.

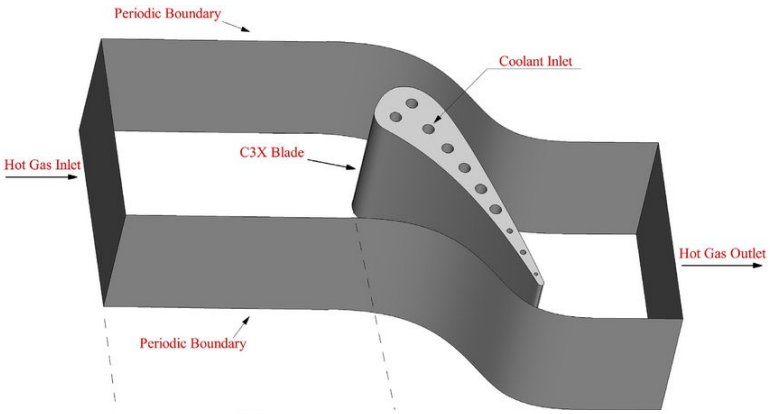


Figure 6.8: *The flow domain of the blade. From [60]. Note that the run inhere was 2-D.*

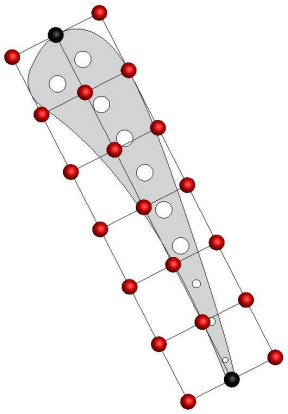


Figure 6.9: *The CPs of the volumetric NURBS parameterizing the blade. CPs at the leading and trailing edges were fixed (black). From [50].*

The network was trained to predict the temperature distribution along the contour of the blade by presenting it with the contour coordinates (x, y) and the distribution of heatflux. Since the number of FSI nodes was equal to 446, the input comprised $3 \times 446 = 1338$ values. The training cost was $\sim 30min$ on the V100 GPU, as well.

Results and K-Fold Cross-Validation

The λ-DNN was trained in the manner of replicating the heat conduction solver in every CHT cycle. Two geometries, not seen by both networks, were used for validating their prediction capabilities. One of them is shown in fig. 6.10. The average MAPE was 0.18% for the modified λ-DNN 0.30% for the λ-DNN presented in the previous cases, and 1.52% for the FCNN proving the superiority of the λ-DNN.

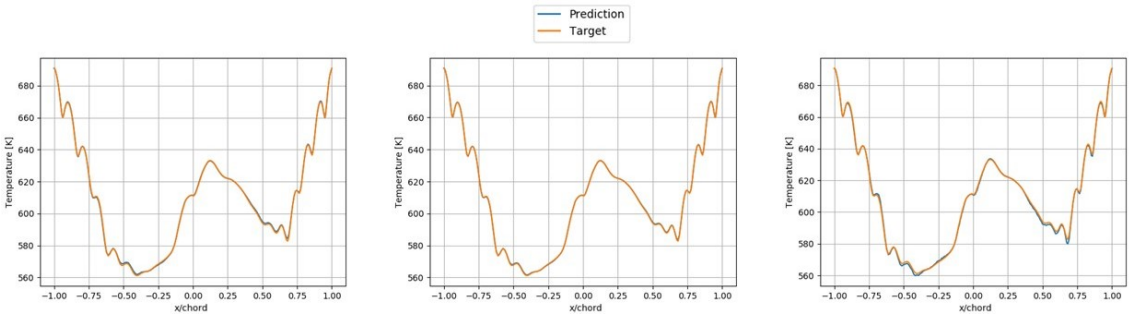


Figure 6.10: *The temperature distribution (prediction) of one, not seen by the networks, blade. The results from the λ-DNN (previous cases) (left), the modified λ-DNN from [50] (middle) and from the FCNN (right).*

In order to validate that the network performance was not influenced by the selection of the training and validation dataset, a 16-fold cross-validation, [61], was performed. The dataset of 160 geometries was split into 16 folds with 10 patterns each. For each unique group of patterns (fold), a fold was taken as a validation dataset. The remaining folds were utilized during the network training as training patterns. Its performance was evaluated on the validation dataset each time by computing the MAPE and, then, the model was discarded. The total MAPE of the cross-validation procedure was computed by averaging the MAPE computed for each fold. The mean value of the computed MAPEs was 0.19% with standard deviation 0.024%. Considering that only 150 geometries were used for the training, the results showcase the capabilities of the network and the randomness in the selection of the training patterns.

Optimization

In order to validate the prediction capabilities of the modified λ-DNN and its ability to replicate the heat solver of the CHT analysis, an optimization procedure was conducted by another author of [50]. The modified λ-DNN is referred as λ-DNN for the optimization section. It is presented shortly, only for demonstrating the current work (DNN training). Further description of this optimization procedure can be found in [50]. The in-house stochastic evolutionary algorithm software, EASY, [62], was utilized for the optimization.

Two optimization objectives were selected. The goals of the optimization were to minimize the mass-averaged total pressure losses between the inlet and outlet of the fluid domain (F_1) and, to minimize the maximum temperature on the solid blade (F_2). Two runs were performed. In the first run (Run_1), λ-DNN acted as a surrogate model for the heat conduction equation solver, providing the fluid solver with the temperature distributions along the blade contour. In the second run (Run_2), the CHT analysis was conducted exclusively on the PUMA software. The total computational cost for both optimization runs was the same. The design variables were

the 19 varying CPs, parameterizing the blade contour. The optimization results, fig. 6.11, showcase that the λ-DNN was capable of replicating the heat equation solver and cooperating with the PUMA RANS solver for optimizing the blade design. The Run_1 front is dominant to the Run_2 one.

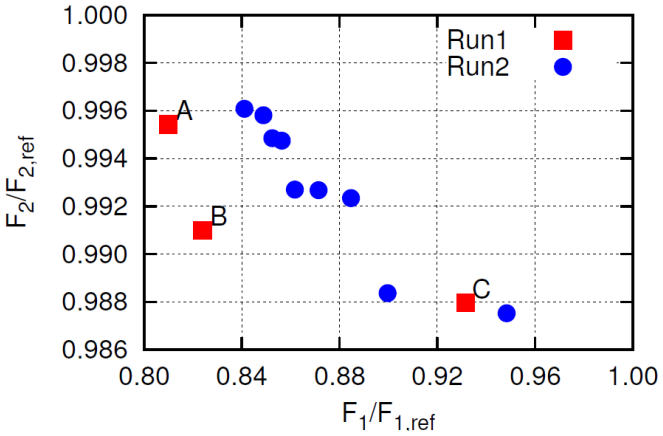


Figure 6.11: *The front of non-dominated solutions of the two runs. From [50].*

Chapter 7

Conclusion

7.1 Overview

In this diploma thesis, two types of DNNs, the LSTM network and the λ-DNN were utilized for predicting flows.

A CFD software, referred as 1DAS, was created, for computing quasi-1D time-varying blood flows. A 1D human artery was modelled with varied initial artery shapes and wall thickness distributions along it. The software, by taking these distributions into account and other artery and blood parameters, numerically solved a system of equations and generated velocity, cross-sectional area and pressure time-varying longitudinal distributions.

Before utilizing the LSTM networks for predicting biological flows, two benchmark cases were perfomed for validating their capabilities in predicting periodic and time-varying distributions. The concept was that the network could predict the following value of a function by presenting it with preceding ones. The networks, accurately, reconstructed the desired distribution with presenting it with the previous values. In fact, in the first case, the network was presented, in addition to the preceding values, with the values of parameters that varied between each training pattern (i.e. parameterization of distribution to be predicted).

Then, the network was called to reconstruct the velocity time-varying distributions in the quasi-1D arterial flow problem. A simplified blood inflow time-series was utilized, in the first case, and later was replaced by a realistic one. In each case, the initial cross-sectional area and the wall thickness distributions varied along the artery, and the training patterns were generated. Note that the network fed back its predictions to itself and utilized them as input for future predictions. The results showcased that the LSTM network is capable of reconstructing velocity distributions

along the artery by presenting it with preceding distributions of the same quantity. An optimization of the LSTM network input and architecture was performed and, thus, the training cost was decreased, without affecting the accurate predictions.

Lastly, another type of DNN, the λ-DNN was used for predicting flows. In the first case, the network reconstructed the pressure field around an isolated airfoil by predicting the pressure value at each node. The network was presented with the nodal coordinates and the coordinates of the airfoil contour. In the second case, the network reconstructed the pressure distribution on the surface of a France runner. The input of the network consisted of the nodal coordinates and the parameters that define the geometry. In the third and last case, the network was utilized in a multi-disciplinary problem, the CHT problem, based on an internally cooled blade. The goal of the λ-DNN was to replicate the heat equation solver in the CHT analysis. In each cycle, the network was presented with the blade contour coordinates and the heatflux distribution along its contour and, predicted the temperature distribution along the same contour. An optimization, that utilized the λ-DNN predictions, was presented, as well.

7.2 Conclusions

By completing the studies conducted in this diploma thesis, the following conclusions are drawn:

1. Both types of DNNs, used to predict flow field (by replacing CFD runs), are capable of making accurate predictions.

2. The LSTM networks are capable of predicting sequential data, demonstrated in the case of biological time-varying flows. They can extract information contained in the preceding values or distributions of a quantity and make accurate predictions of the following ones. Since they may feed back their prediction to themselves, they utilize very limited pieces of information. In the biological flows, the initial distributions provided to the LSTM network include the information of the varying initial arterial shape and the wall thickness. Since the blood inflow is constant in each training pattern, there is no need to present the network with more information about the parameterization. Their main advantage is that their prediction depends exclusively on the previous states of the predicting quantity. Statistical methods can be used to optimize the network input consisting of sequential data and, with the addition of architecture optimization, the training cost can be further decreased.

3. The λ-DNN is capable of predicting aerodynamic flows. The network reconstructs, accurately, the entire fields, by making predictions node by node, and by presenting it with the coordinates and case related data. The low number of flows used as training patterns showcase the accurate performance of this

architecture.

4. The λ-DNN architecture is capable of replicating a disciplines into a multi-disciplinary analysis, CHT, as well. The network predicts the entire temperature field by providing with the corresponding input. In addition to the accurate predictions, the network is exploited in an optimization procedure, with a stochastic evolutionary algorithm. The front of the optimization using the λ-DNN dominates the one which used the CHT software. Thus, the λ-DNN architecture can be exploited in CHT optimization by replicating one discipline and, thus, decreasing the total cost of the procedure.

7.3 Future Work Proposals

Based on the implementation of the DNNs in CFD, the following future works are proposed:

1. The LSTM can further be incorporated in predicting biological flows. A more expensive dataset can be generated (i.e. time- and resources-consuming data generation), containing data from 3D flows. Since the network yields high prediction of sequential data with little information about the case related data, it can be further exploited in micro-scale study of human deceases.
2. The LSTM can further be incorporated in other CFD problems, as well. In unsteady runs of CFD software, the demand of computational resources is very high. Thus, the LSTM network can be utilized in predicting unsteady flows, by presenting it with the preceding ones.
3. More types of DNNs can be exploited in predicting CFD flows. For instance, by incorporating the λ-DNN into a GAN network, the predictions could be further improved in accuracy. In fact, λ-DNN could replace the typical generator of a GAN network.
4. The λ-DNN can replicate other, or more, disciplines, in a multi-disciplinary problem.

Bibliography

[1] Wu, Y., , and et al.: *Google's neural machine translation system: Bridging the gap between human and machine translation.* CoRR, abs/1609.08144, 2016. http://arxiv.org/abs/1609.08144.

[2] Apple Inc.: *Using apple watch for arrhythmia detection*, 2020. https://www.apple.com/healthcare/docs/site/Apple_Watch_Arrhythmia_Detection.pdf.

[3] Bhalla, A., Nikhila, M. S., and Singh, P.: *Simulation of self-driving car using deep learning.* In *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, pages 519–525, 2020.

[4] Daley, J.: *A.I. Mastered Backgammon, Chess and Go. Now It Takes On StarCraft II*, October 2019. https://bit.ly/2SglD4d, [Online; accessed 01-June-2021].

[5] Whitwam, R.: *DeepMind AI Challenges Pro StarCraft II Players, Wins Almost Every Match*, January 2019. https://bit.ly/3w01DBM, [Online; accessed 01-June-2021].

[6] Vinyals, O. and et al.: *Grandmaster level in StarCraft II using multi-agent reinforcement learning.* Nature, 575(7782):350–354, October 2019. https://doi.org/10.1038/s41586-019-1724-z, 10.1038/s41586-019-1724-z.

[7] Ben Khalifa, R., Yahia, N., and Zghal, A.: *Integrated neural networks approach in cad/cam environment for automated machine tools selection.* Journal of Mechanical Engineering Research, 2:25–38, April 2010.

[8] Samanta, B.: *Gear fault detection using artificial neural networks and support vector machines with genetic algorithms.* Mechanical Systems and Signal Processing, 18(3):625–644, 2004, ISSN 0888-3270. https://www.sciencedirect.com/science/article/pii/S0888327003000207, https://doi.org/10.1016/S0888-3270(03)00020-7.

[9] Hosoz, M., Ertunc, H.M., and Bulgurcu, H.: *Performance prediction of a cooling tower using artificial neural network.* Energy Conversion and Management, 48(4):1349–1359, 2007, ISSN 0196-8904. https:

//www.sciencedirect.com/science/article/pii/S0196890406003165, https://doi.org/10.1016/j.enconman.2006.06.024.

[10] Goodfellow, I., Bengio, Y., and Courville, A.: *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[11] Schmidhuber, J.: *Deep learning in neural networks: An overview*. Neural Networks, 61:85–117, January 2015. https://doi.org/10.1016/j.neunet.2014.09.003, 10.1016/j.neunet.2014.09.003.

[12] Wang, B and Wang, J.: *Application of artificial intelligence in computational fluid dynamics*. Industrial & Engineering Chemistry Research, 60(7):2772–2790, 2021. https://doi.org/10.1021/acs.iecr.0c05045, 10.1021/acs.iecr.0c05045.

[13] Guo, X., Li, W., and Iorio, F.: *Convolutional neural networks for steady flow approximation*. In *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 481–490, New York, USA, 2016.

[14] Li, Y., Chang, J., Kong, C., and Wang, Z.: *Flow field reconstruction and prediction of the supersonic cascade channel based on a symmetry neural network under complex and variable conditions*. AIP Adv., 10(6):065116, 2020.

[15] Bhatnagar, S., Afshar, Y., Pan, S., Duraisamy, K., and Kaushik, S.: *Prediction of aerodynamic flow fields using convolutioanl neural networks*. 64:525–545, 2019.

[16] Wu, H., Liu, X., An, W., Chen, S., and Lyu, H.: *A deep learning approach for efficiently and accurately evaluating the flow field of supercritical airfoils*. Computers & Fluids, 198:104393, 2020, ISSN 0045-7930. https://doi.org/10.1016/j.compfluid.2019.104393.

[17] Giannakoglou, K.: *Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence*. Prog. Aerosp. Sci., 38(1):43–76, 2002.

[18] Karakasis, M. and Giannakoglou, K.: *On the use of metamodel-assisted, multiobjective evolutionary algorithms*. Eng. Optim., 38(8):941–957, 2006.

[19] Kapsoulis, D.: *Low-Cost Metamodel-Assisted Evolutionary Algorithms with Application in Shape Optimization in Fluid Dynamics*. PhD thesis, National Technical University of Athens, Athens, 2020.

[20] Krizhevsky, A., Sutskever, I., and Hinton, G. E.: *ImageNet classification with deep convolutional neural networks*. Communications of the ACM, 60(6):84–90, May 2017. https://doi.org/10.1145/3065386, 10.1145/3065386.

[21] Passricha, V. and Aggarwal, R. K.: *Convolutional neural networks for raw speech recognition*. In *From Natural to Artificial Intelligence - Algorithms*

and Applications. IntechOpen, December 2018. https://doi.org/10.5772/intechopen.80026.

[22] Amato, F. and et al.: *Artificial neural networks in medical diagnosis*. J Appl Biomed, 11:47–58, December 2013. 10.2478/v10136-012-0031-x.

[23] Gatys, L. A., Ecker, A. S., and Bethge, M.: *A neural algorithm of artistic style*, 2015.

[24] Haykin, S.: *Neural networks and learning machines*. Prentice Hall/Pearson, New York, 2009, ISBN 978-0131471399.

[25] Rumelhart, D. E., Hinton, G. E., and Williams, R. J.: *Learning representations by back-propagating errors*. Nature, 323(6088):533–536, October 1986. https://doi.org/10.1038/323533a0, 10.1038/323533a0.

[26] Kingma, D. P. and Ba, J.: *Adam: A method for stochastic optimization*. In Bengio, Yoshua and LeCun, Yann (editors): *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. http://arxiv.org/abs/1412.6980.

[27] Duchi, J., Hazan, E., and Singer, Y.: *Adaptive subgradient methods for online learning and stochastic optimization*. Journal of Machine Learning Research, 12:2121–2159, July 2011.

[28] Tieleman, T. and Hinton, G.: *Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude*. COURSERA: Neural Networks for Machine Learning, 2012.

[29] Hochreiter, S. and Schmidhuber, J.: *Long short-term memory*. Neural Computation, 9(8):1735–1780, November 1997. https://doi.org/10.1162/neco.1997.9.8.1735, 10.1162/neco.1997.9.8.1735.

[30] Sak, H., Senior, A., and Beaufays, F.: *Long short-term memory recurrent neural network architectures for large scale acoustic modeling*. Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, pages 338–342, January 2014.

[31] Graves, A. and et al.: *A novel connectionist system for unconstrained handwriting recognition*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(5):855–868, 2009. 10.1109/TPAMI.2008.137.

[32] Abadi, M. and et al.: *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015. http://tensorflow.org/, Software available from tensorflow.org.

[33] Sherwin, S.J., Franke, V., Peiró, J., and Parker, K.: *One-dimensional modelling of a vascular network in space-time variables*. Journal of Engineering Mathematics, 47(3/4):217–250, 2003, ISSN 0022-0833. 10.1023/b:engi.0000007979.32871.e2.

[34] Raines, J. K., Jaffrin, M. Y., and Shapiro, A. H.: *A computer simulation of arterial dynamics in the human leg.* Journal of Biomechanics, 7(1):77–91, January 1974. https://doi.org/10.1016/0021-9290(74)90072-4, 10.1016/0021-9290(74)90072-4.

[35] Quarteroni, A. and Formaggia, L.: *Mathematical modelling and numerical simulation of the cardiovascular system.* In *Computational Models for the Human Body*, volume 12 of *Handbook of Numerical Analysis*, pages 3–127. Elsevier, 2004. https://www.sciencedirect.com/science/article/pii/S1570865903120017.

[36] Shapiro, A. H.: *Steady flow in collapsible tubes.* Journal of Biomechanical Engineering, 99(3):126–147, August 1977. https://doi.org/10.1115/1.3426281, 10.1115/1.3426281.

[37] Reuderink, P.J., Hoogstraten, H.W., Sipkema, P., Hillen, B., and Westerhof, N.: *Linear and nonlinear one-dimensional models of pulse wave transmission at high womersley numbers.* Journal of Biomechanics, 22(8-9):819–827, January 1989. https://doi.org/10.1016/0021-9290(89)90065-1, 10.1016/0021-9290(89)90065-1.

[38] Hirsch, Ch: *Numerical computation of internal and external flows.* Wiley, Chichester England New York, 1988, ISBN 978-0-471-92452-4.

[39] Westerhof, N., Lankhaar, J., and Westerhof, B. E.: *The arterial windkessel.* Medical & Biological Engineering & Computing, 47(2):131–141, June 2008. https://doi.org/10.1007/s11517-008-0359-2, 10.1007/s11517-008-0359-2.

[40] Heller, D.: *Direct and iterative methods for block tridiagonal linear systems.* 1977.

[41] Anliker, M., Rockwell, R. L., and Ogden, E.: *Nonlinear analysis of flow pulses and shock waves in arteries.* Zeitschrift für angewandte Mathematik und Physik ZAMP, 22(2):217–246, March 1971. https://doi.org/10.1007/bf01591407, 10.1007/bf01591407.

[42] Engineering ToolBox: *Dynamic viscosity of some common liquids.*, 2008. https://www.engineeringtoolbox.com/absolute-viscosity-liquids-d_1259.html, [Online; accessed 01-March-2021].

[43] Maas, A. L., Hannun, A. Y., and Ng, A. Y.: *Rectifier nonlinearities improve neural network acoustic models.* In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[44] Nowak, M. and et al.: *The protocol for using elastic wall model in modeling blood flow within human artery.* European Journal of Mechanics - B/Fluids, 77:273–280, September 2019. https://doi.org/10.1016/j.euromechflu.2019.03.009, 10.1016/j.euromechflu.2019.03.009.

[45] Chetlur, S. and et al.: *cudnn: Efficient primitives for deep learning.* CoRR, abs/1410.0759, 2014. http://arxiv.org/abs/1410.0759.

[46] Guthrie, W. F.: *NIST/SEMATECH e-Handbook of Statistical Methods (NIST Handbook 151).* National Institute of Standards and Technology, 2020. https://www.itl.nist.gov/div898/handbook/.

[47] Hertel, L. and et al.: *Sherpa: Robust hyperparameter optimization for machine learning.* SoftwareX, 2020. In press.

[48] Jaderberg, M. and et al.: *Population based training of neural networks*, 2017.

[49] Kontou, M., Kapsoulis, D., Baklagis, I., and Giannakoglou, K.: *λ-DNNs and Their Implementation in Aerodynamic and Conjugate Heat Transfer Optimization.* In *Proceedings of the 21st EANN (Engineering Applications of Neural Networks) 2020 Conference*, pages 202–214. Springer International Publishing, May 2020, ISBN 978-3-030-48790-4.

[50] Kontou, M., Kapsoulis, D., Baklagis, I., Trompoukis, X., and Giannakoglou, K.: *λ-DNNs and their implementation in conjugate heat transfer shape optimization.* Neural Computing and Applications, March 2021. https://doi.org/10.1007/s00521-021-05858-2, 10.1007/s00521-021-05858-2.

[51] Aslani, S. and et al.: *Multi-branch convolutional neural network for multiple sclerosis lesion segmentation.* NeuroImage, 196:1–15, 2019, ISSN 1053-8119. https://www.sciencedirect.com/science/article/pii/S105381191930268X, https://doi.org/10.1016/j.neuroimage.2019.03.068.

[52] Huu, T., Nguyen, D., Tsiligianni, E., Cornelis, B., and Deligiannis, N.: *Multi-view deep learning for predicting twitter users' location.* December 2017.

[53] Hennigh, O.: *Automated design using neural networks and gradient descent.* October 2017.

[54] Guo, X., Li, W., and Iorio, F.: *Convolutional neural networks for steady flow approximation.* In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, August 2016. https://doi.org/10.1145/2939672.2939738.

[55] Zhang, H., Shao, J., and Salakhutdinov, R.: *Deep neural networks with multi-branch architectures are intrinsically less non-convex.* In *AISTATS*, 2019.

[56] Kampolis, I., Trompoukis, X., Asouti, V., and Giannakoglou, K.: *CFD–based analysis and two–level aerodynamic optimization on Graphics Processing Units.* Computer Methods in Applied Mechanics and Engineering, 199(9–12):712–722, 2010.

[57] Tsiakas, K.T., Gagliardi, F., Trompoukis, X.S., and Giannakoglou, K.C.: *Shape optimization of turbomachinery rows using a parametric blade modeller and the continuous adjoint method running on GPUS.* In *ECCOMAS Congress 2016,*

VII European Congress on Computational Methods in Applied Sciences and Engineering, Crete, Greece, June 5-10 2016.

[58] Hylton, L. D. and et al.: *Analytical and experimental evaluation of the heat transfer distribution over the surfaces of turbine vanes.* Technical report 19830020105, National Aeronautics and Space Administration (NASA), 1983.

[59] Menter, F.R., Kuntz, M., and Langtry, R.: *Ten years of industrial experience with SST turbulence model.* Heat Mass Transf., 4:625–632, 2003.

[60] Karimi, M. S. and et al.: *Robust optimization of the nasa c3x gas turbine vane under uncertain operational conditions.* International Journal of Heat and Mass Transfer, 164:120537, 2021, ISSN 0017-9310. 10.1016/j.ijheatmasstransfer.2020.120537.

[61] Hastie, T., Tibshirani, R., and Friedman, J.: *The elements of statistical learning, second edition: data mining, inference, and prediction.* Springer, 2016.

[62] The EASY (Evolutionary Algorithms SYstem) software, http://velos0.ltt.mech.ntua.gr/EASY., 2008.

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

Πρόβλεψη Ροών με Βαθιά Νευρωνικά Δίκτυα

Εκτενής Περίληψη Διπλωματικής Εργασίας

Ιωάννης Μπακλαγής

Επιβλέπων
Κυριάκος Χ. Γιαννάκογλου, Καθηγητής ΕΜΠ

Αθήνα, 2021

Τεχνητή Νοημοσύνη και Βαθιά Νευρωνικά Δίκτυα

Τις τελευταίες δεκαετίες ο τομέας της Τεχνητής Νοημοσύνης (ΤΝ) έχει γνωρίσει μία ραγδαία ανάπτυξη εισχωρώντας βαθιά τόσο στην καθημερινή ζωή του ανθρώπου όσο και στους βιομηχανικούς και ακαδημαϊκούς τομείς. Η ΤΝ έχει τη δυνατότητα να ανιχνεύει μοτίβα μέσα σε πληθώρα δεδομένων, να λαμβάνει αποφάσεις και να επιλύει δύσκολα προβλήματα εκτελώντας διάφορες διαδικασίες.

Η Μηχανική Μάθηση (ΜΜ) ανήκει στο πεδίο της ΤΝ. Τα μοντέλα ΜΜ έχουν τη δυνατότητα να ΄μαθαίνουν΄ αυτόματα όταν τροφοδοτούνται με δεδομένα. Τα Νευρωνικά Δίκτυα (ΝΔ), είναι μοντέλα ΜΜ, τα οποία αποτελούνται από διασυνδεδεμένα δίκτυα τεχνητών νευρώνων. Αυτά μπορούν να εκπαιδευτούν να κάνουν προβλέψεις σε δεδομένα τροφοδοτώντας τα με την αντίστοιχη είσοδο και ελαχιστοποιώντας το σφάλμα μεταξύ της εξόδου τους και της πραγματικής/ιδανικής εξόδου. Τα ΝΔ έχουν χρησιμοποιηθεί ευρέως και στον τομέα της Υπολογιστικής Ρευστοδυναμικής (ΥΡΔ), αξιοποιώντας τα, ώστε να κάνουν προβλέψεις σε δεδομένα προερχόμενα από λογισμικά ΥΡΔ. Εξαιτίας του μικρού υπολογιστικού κόστος των προβλέψεων τους, τα ΝΔ χρησιμοποιούνται επίσης και ως υποκατάστατα των λογισμικών ΥΡΔ σε διαδικασίες βελτιστοποίησης.

Η βασική μονάδα ενός ΝΔ είναι ο νευρώνας, ο οποίος είναι μία μαθηματική σχέση, δέχεται σήμα από άλλους νευρώνες, το επεξεργάζεται και σηματοδοτεί τους επόμενους. Τα σήματα πολλαπλασιάζονται με ένα συναπτικό βάρος. Οι νευρώνες ενός ΝΔ οργανώνονται σε επίπεδα. Όταν τα ΝΔ έχουν παραπάνω από 2 κρυφά επίπεδα, αυτά ονομάζονται Βαθιά Νευρωνικά Δίκτυα (ΒΝΔ). Οι δύο βασικοί παράγοντες της αρχιτεκτονικής του ΝΔ είναι το πλήθος των επιπέδων και των νευρώνων ανά επίπεδο. Τα δίκτυα εκπαιδεύονται μέσω ενός αλγορίθμου μάθησης, ο οποίος μεταβάλλει τα συναπτικά βάρη. Υπάρχουν 2 είδη δικτύων, τα πρόσθιας τροφοδότησης (feedforward), στα οποία η πληροφορία έχει κατεύθυνση από την είσοδο προς την έξοδο και τα ανατροφοδοτούμενα (recurrent), τα οποία ανατροφοδοτούν την έξοδο τους στην είσοδο. Τα δίκτυα που χρησιμοποιήθηκαν στην διπλωματική εργασία είναι το λ-DNN, το οποίο εμπίπτει στην πρώτη κατηγορία και το LSTM, το οποίο εμπίπτει στη δεύτερη.

Ψευδο-Μονοδιάστατη Ροή Αίματος σε Αρτηρία

Στο κεφάλαιο αυτό, μοντελοποιείται μια ψευδο-μονοδιάστατη ανθρώπινη αρτηρία με ελαστικά τοιχώματα. Το λογισμικό που προγραμματίζεται χρησιμοποιείται ώστε να συλλεγθούν δείγματα τα οποία θα εκπαιδεύσουν τα ΝΔ. Οι εξισώσεις ορμής και συνέχειας και κλίσης πίεσης δίνουν ένα σύστημα τριών αγνώστων, της ταχύτητας (u), του εμβαδού της διατομής (A) και της πίεσης (p),

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = \mathbf{S} \tag{1}$$

όπου

$$\mathbf{U} = \begin{bmatrix} A \\ u \end{bmatrix},\ \mathbf{F} = \begin{bmatrix} uA \\ \frac{u^2}{2} + \frac{p}{\rho} \end{bmatrix} and\ \mathbf{S} = \begin{bmatrix} 0 \\ -\frac{8\pi\mu u}{\rho A} \end{bmatrix}$$

όπου ρ και μ είναι η σταθερή πυκνότητα και η συνεκτικότητα του αίματος, αντίστοιχα.

Επειδή υπάρχουν 2 εξισώσεις και 3 άγνωστοι (ταχύτητα, πίεση και εμβαδόν διατομής) απαιτείται και μία ακόμη εξίσωση. Αυτή η εξίσωση δείχνει τη σχέση μεταξύ της πίεσης στην αρτηρία με το εμβαδό της διατομής, την πίεση γύρω από την αρτηρία (p_{ext}) και το εμβαδό στο οποίο εσωτερικά και εξωτερικά υπάρχει ισορροπία της πίεσης (A_0).

$$p = p_{ext} + \beta(\sqrt{A} - \sqrt{A_0}) \tag{2}$$

Το β χαρακτηρίζεται από τις μηχανικές ιδιότητες του ελαστικού τοιχώματος,

$$\beta = \frac{\sqrt{\pi}hE}{A_0(1-\sigma^2)} \tag{3}$$

όπου h είναι το πάχος του αρτηριακού τοιχώματος, E το μέτρο ελαστικότητας και σ ο λόγος Poisson. Η πίεση στο σύστημα, εξ.(1), αντικαθίσταται από την εξ.(2) και προκύπτει το σύστημα,

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{H}\frac{\partial \mathbf{U}}{\partial x} = \mathbf{C} \tag{4}$$

όπου

$$\mathbf{U} = \begin{bmatrix} A \\ u \end{bmatrix},\ \mathbf{H} = \begin{bmatrix} u & A \\ \frac{\beta}{2\rho\sqrt{A}} & u \end{bmatrix}$$

$$\mathbf{C} = -\frac{1}{\rho}\begin{bmatrix} 0 \\ \frac{\mu^*\rho u}{A} + \frac{\partial p_{ext}}{\partial x} - \frac{\beta}{2\sqrt{A_0}}\frac{\partial A_0}{\partial x} + (\sqrt{A} - \sqrt{A_0})\frac{\partial \beta}{\partial x} \end{bmatrix}$$

Εφαρμόζεται η τεχνική του Flux Vector Splitting, έπειτα το σύστημα διακριτοποιείται και προκύπτει ένα πενταδιαγώνιο σύστημα εξισώσεων (για ακρίβεια δεύτερης τάξης) το οποίο επιλύεται αριθμητικά.

Η πίεση στην έξοδο της αρτηρίας ορίζεται από το μοντέλο Windkessel το οποίο λαμβάνει υπόψη την επίδραση των υπόλοιπων αρτηριών, πέρα αυτής που αναλύεται. Στα άκρα της αρτηρίας υπάρχουν οριακές συνθήκες και το πεδίο αρχικοποιείται κατάλληλα. Έτσι, δημιουργείται το λογισμικό 1DAS, το οποίο υπολογίζει τη ροή και παράγει χρονομεταβλητές χωρικές κατανομές ταχύτητας, πίεσης και εμβαδού. Επίσης δημιουργήθηκε και μία εκδοχή του λογισμικού, το οποίο μοντελοποιεί αρτηρίες με άκαμπτα τοιχώματα.

Εφαρμογές με LTSM

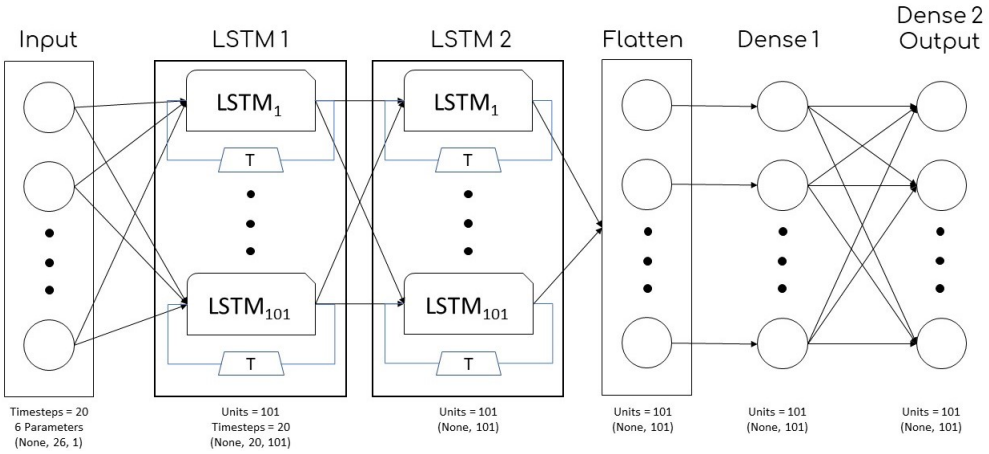
Πριν την κύρια εφαρμογή των LSTM στην πρόβλεψη βιολογικών ροών, παρουσιάζονται 2 απλές εφαρμογές, οι οποίες καταδεικνύουν τις δυνατότητές τους.

Απλή Περιοδική Συνάρτηση

Για τη δημιουργία δειγμάτων εκπαίδευσης χρησιμοποιείται μία απλή περιοδική συνάρτηση,

$$y = a_1 sin(a_2 x) + a_3 cos(a_4 x) + sin(a_5 x) + a_6 \tag{5}$$

όπου $x \in [0, 160]$ αναπαριστά τον ψευδό-χρόνο και $a_1, ..., a_6 \in [0, 1]$ είναι παράμετροι. Μεταβάλλοντας τις παραμέτρους δημιουργούνται δεδομένα εκπαίδευσης. Ο x άξονας διακριτοποιείται με 401 κόμβους, σε σταθερά διαστήματα. Έχουν παραχθεί 28 χρονοσειρές και το υπολογιστικό κόστος ήταν $\sim 5mins$. Σκοπός του δικτύου είναι να προβλέψει την επόμενη τιμή της συνάρτησης, τροφοδοτώντας το με προηγούμενες. Η αρχιτεκτονική του δικτύου παρουσιάζεται στο Σχ. 1 και έχει ως είσοδο τις προηγούμενες 20 τιμές της συνάρτησης καθώς και τις τιμές των 6 παραμέτρων. Τα αποτελέσματα είναι πολύ ικανοποιητικά και αξιολογούνται σε 2 συναρτήσεις τις οποίες δεν είχε δει το δίκτυο κατά τη διάρκεια της εκπαίδευσης, Σχ. 2, με μέσο MAE σφάλμα 3×10^{-2}.



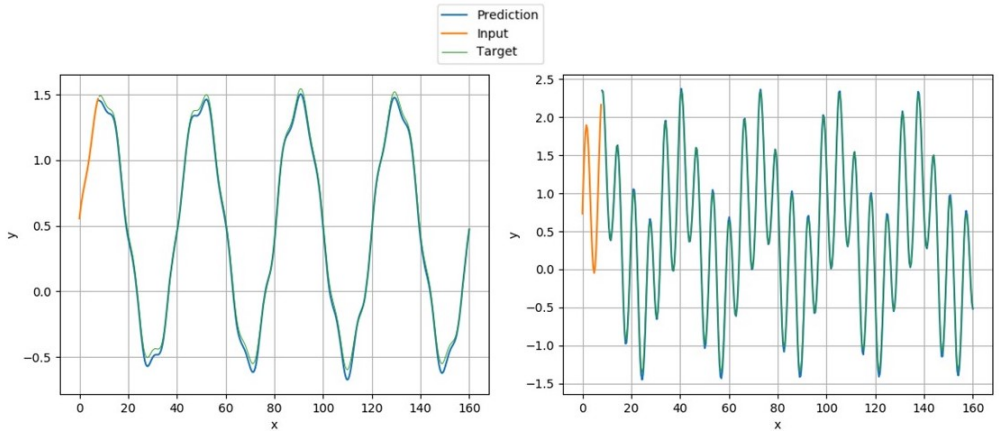
Σχήμα 1: *Η αρχιτεκτονική του LSTM δικτύου.*

Εξίσωση Αγωγής Θερμότητας

Σε αυτήν την εφαρμογή το δίκτυο καλείται να προβλέψει την κατανομή θερμοκρασίας μίας πλάκας, τροφοδοτώντας το με κατανομές σε προηγούμενες χρονικές στιγμές. Η εξίσωση που διέπει την αγωγή θερμότητας είναι

$$\frac{\partial}{\partial x}(k\frac{\partial T}{\partial x}) = \rho C_{pl}\frac{\partial T}{\partial t} \tag{6}$$

όπου T η θερμοκρασία της πλάκας, $k = 2W/m^{\circ}C$ ο συντελεστής θερμικής αγωγι-



Σχήμα 2: *Οι προβλέψεις του δικτύου σε 2 συναρτήσεις τις οποίες δεν είδε το δίκτυο κατά την εκπαίδευσή του.*

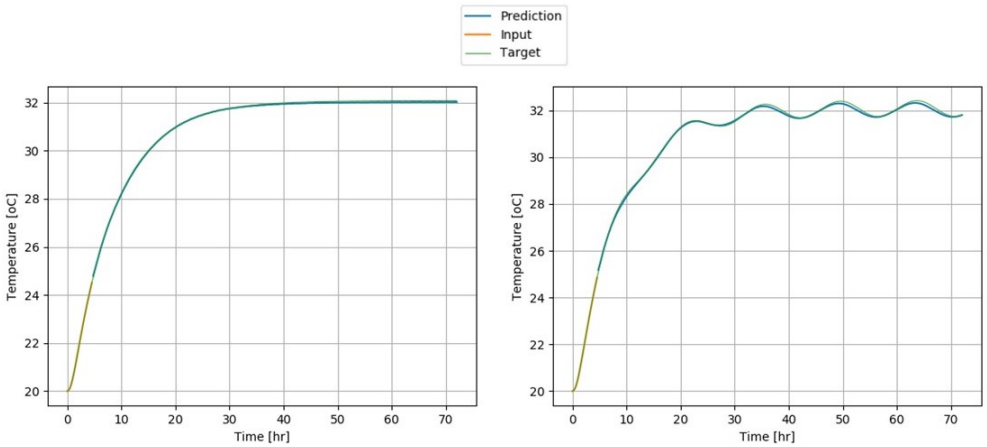
μότητας, $\rho = 2000kg/m^3$ η πυκνότητα και, $C_{pl} = 1000J/kg°C$ ο συντελεστής θερμοχωρητικότητας. Στο δεξί άκρο υπάρχει αέρας θερμοκρασίας $T_{in} = 27°C$ ενώ στο αριστερό η θερμοκρασία του αέρα είναι περιοδική (T_{out}),

$$T_{out} = 35 + b_1 cos(\frac{2\pi(t - 3600b_2)}{3600b_3}) \tag{7}$$

όπου $b_1, b_2 \in [0, 24]$ και $b_3 \in [-5, 5]$ είναι παράμετροι. Η εξίσωση επιλύεται με τη μέθοδο των πεπερασμένων όγκων και προκύπτουν χρονομεταβαλλόμενες χωρικές κατανομές θερμοκρασίας της πλάκας. Μεταβάλλοντας τις παραμέτρους, δημιουργούνται διάφορες κατανομές και προέκυψε μία βάση δεδομένων για την εκπαίδευση του LSTM δικτύου. Το δίκτυο, με την αρχιτεκτονική της προηγούμενης εφαρμογής τροποποιημένη, εκπαιδεύεται με 20 διαφορετικές κατανομές T_{out} και καλείται να προβλέψει την επόμενη κατανομή (59 κελιά άρα και 59 θερμοκρασίες) τροφοδοτώντας το με 20 κατανομές. Το υπολογιστικό κόστος της εκπαίδευσης είναι $\sim 5mins$. Τα αποτελέσματα του δικτύου αξιολογούνται από 2 κατανομές τις οποίες δεν έχει δει και κρίνονται πολύ ικανοποιητικά, Σχ. 3, αφού τροφοδοτείται μόνο με τις 20 κατανομές, χωρίς καμία πληροφορία για τις μεταβλητές b. Το μέσο MAE σφάλμα είναι $15 \times 10^{-2}\ ^oC$.

Πρόβλεψη 1Δ Χρονομεταβλητών Ροών σε Αρτηρίες με LTSM Δίκτυα

Στη διπλωματική εργασία, πραγματοποιείται μία αρχική απλοποιημένη μελέτη στο πώς τα LSTM μπορούν να εφαρμοστούν στην έρευνα βιολογικών ροών. Το LSTM δίκτυο εκπαιδεύεται να προβλέπει τις επόμενες κατανομές ταχύτητας κατά μήκος της αρτηρίας, τροφοδοτούμενο προηγούμενες κατανομές του ίδιου μεγέθους, ως είσοδο. Το λογισμικό 1DAS παρέχει τα δείγματα εκπαίδευσης στο δίκτυο.



Σχήμα 3: *Οι προβλέψεις του LSTM δικτύου σε δύο κατανομές τις οποίες δεν έχει δει το δίκτυο. Όλες οι χρονοσειρές αναφέρονται στο 30ο κελί.*

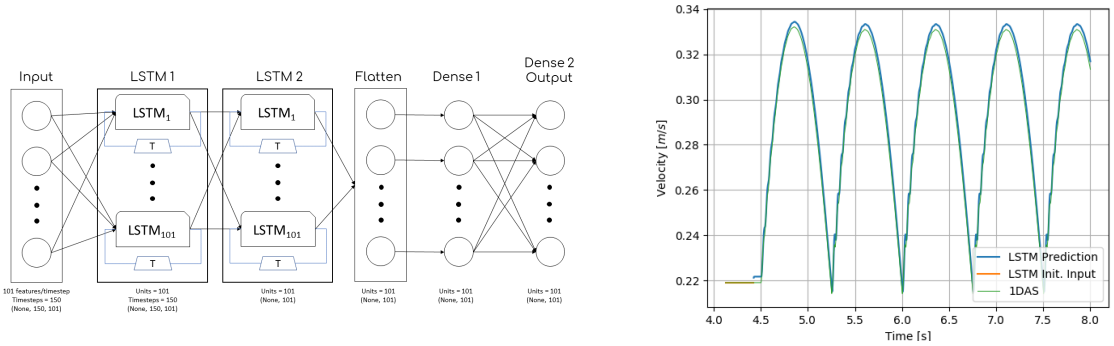
Μεταβαλλόμενα Αρχικά Σχήματα Αρτηρίας

Στην πρώτη εφαρμογή, η κατανομή του αρχικού εμβαδού διατομής της αρτηρίας παραμετροποιείται με μία καμπύλη Bezier με 7 σημεία ελέγχου (ΣΕ). Τα ΣΕ μεταβάλλονται κατά τον y-άξονα και, παράγονται διαφορετικές κατανομές εμβαδών. Η παροχή αίματος είναι χρονομεταβλητή και δίνεται από μία απλή ημιτονοειδή συνάρτηση. Για τα πρώτα $6s$ η παροχή διατηρείται σταθερή, και έπειτα, αυτή μεταβάλλεται περιοδικά. Προέκυψε έτσι μια αρχικά μεταβατική φάση και έπειτα η περιοδική. Ο συνολικός χρόνος προσομοίωσης ήταν $8s$, ο οποίος διακριτοποιήθηκε με 8000 χρονικά βήματα, ενώ το μήκος της αρτηρίας με 101 κόμβους.

Το λογισμικό 1DAS, παίρνοντας ως εισόδους το αρχικό σχήμα της αρτηρίας, τη χρονοσειρά της παροχής αίματος και τα μηχανικά στοιχεία της αρτηρίας, παράγει κατανομές πίεσης, ταχύτητας και εμβαδού. Οι κατανομές που χρησιμοποιούνται ως βάση δεδομένων προήλθαν έπειτα απο δειγματοληψία ανά 20 χρονικά βήματα (375 χρονικά βήματα ανά περίοδο), μετά το τέλος της αρχικής μεταβατικής φάσης. Το δίκτυο για την πρόβλεψη της επόμενης κατανομής της ταχύτητας τροφοδοτείται με είσοδο 150 κατανομές, οι οποίες προέρχονται είτε απο το λογισμικό 1DAS είτε είναι προβλέψεις του ίδιου του δικτύου. Κατά την πρόβλεψη, λοιπόν, των απαιτούμενων κατανομών ταχύτητας τροφοδοτείται αρχικά με 150 κατανομές από το λογισμικό 1DAS. Το δίκτυο προβλέπει την επόμενη κατανομή και έπειτα ανατροφοδοτεί αυτήν ως είσοδο, συνεχίζοντας, έτσι, τις προβλέψεις. Η αρχιτεκτονική παρουσιάζεται στο Σχ. 4. Η εκπαίδευση διήρκησε $\sim 1.6 hours$, σε μία NVIDIA 1050 GPU. Η εκπαίδευση χωρίστηκε σε 3 τμήματα, όπου ανανεωνόταν η βάση δεδομένων με προβλέψεις από το δίκτυο.

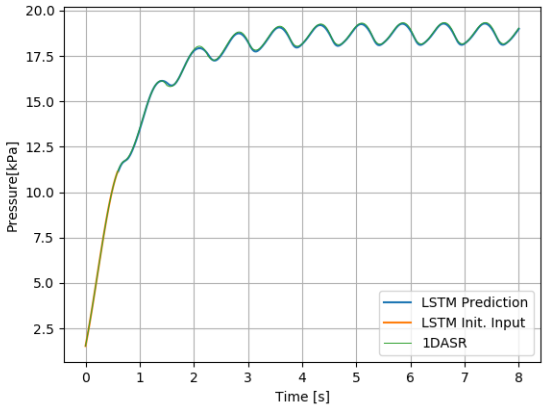
Τα αποτελέσματα του δικτύου είναι πολύ ικανοποιητικά αφού επιτυγχάνει να προβλέψει τις κατανομές με μόνο 150 κατανομές ως αρχική είσοδο, Σχ. 4. Το μέσο MAE σφάλμα των προβλέψεων είναι $6 \times 10^{-3} m/s$.

Αρτηρία με άκαμπτα τοιχώματα



Σχήμα 4: *Η αρχιτεκτονική του δικτύου (αριστερά) και η χρονοσειρά της ταχύτητας στον 50ο κόμβο, ενός δείγματος το οποίο δεν έχει δει το δίκτυο (δεξιά).*

Στην επόμενη φάση, το δίκτυο επίσης εκπαιδεύεται να προβλέπει τις κατανομές ταχύτητες σε αρτηρία με άκαμπτα τοιχώματα. Όλες οι παράμετροι καθώς και οι κατανομές του λογισμικού παρέμειναν ίδιες. Το δίκτυο εκπαιδεύεται και με επιτυχία προβλέπει τις κατανομές ταχύτητας, Σχ. 5, έχοντας μέσο MAE σφάλμα $1.6 \times 10^{-3} m/s$.

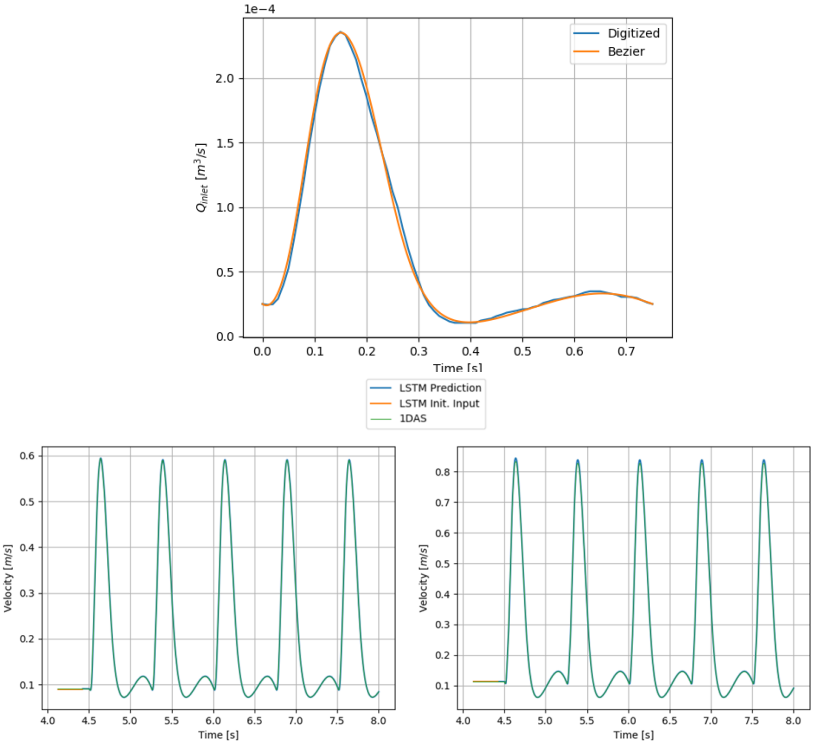


Σχήμα 5: *Η χρονοσειρά της ταχύτητας-πρόβλεψη του δικτύου στο 50ο κόμβο, την οποία το δίκτυο δεν είχε δει κατά τη διάρκεια της εκπαίδευσης*

Πραγματική Παροχή Αίματος

Σε αυτήν την εφαρμογή χρησιμοποιείται μία χρονοσειρά παροχής του αίματος από τη βιβλιογραφία, η οποία αναπαριστά μία πραγματική ροή αίματος, Σχ. 6. Αυτή ψηφιοποιείται και γίνεται παρεμβολή με σκοπό να αξιοποιηθεί από το λογισμικό 1DAS. Χρησιμοποιήθηκε η βιβλιοθήκη της NVIDIA, cuDNN, για την εκπαίδευση του BNΔ και έτσι μεταβάλλεται ελάχιστα η αρχιτεκτονική του δικτύου, μειώνοντας το κόστος εκπαίδευσης σε $\sim 1 hour$. Παρόμοια με την προηγούμενη εφαρμογή, δημιουργείται η βάση δεδομένων και πραγματοποιείται η εκπαίδευση.

Εξαιτίας της αλλαγής στην αρχιτεκτονική του δικτύου, παρατηρείται βελτίωση στα αποτελέσματα/προβλέψεις του, Σχ. 6, συγκριτικά με την προηγούμενη εφαρμογή, μειώνοντας το μέσο σφάλμα MAE σφάλμα σε $1.4 \times 10^{-3} m/s$.



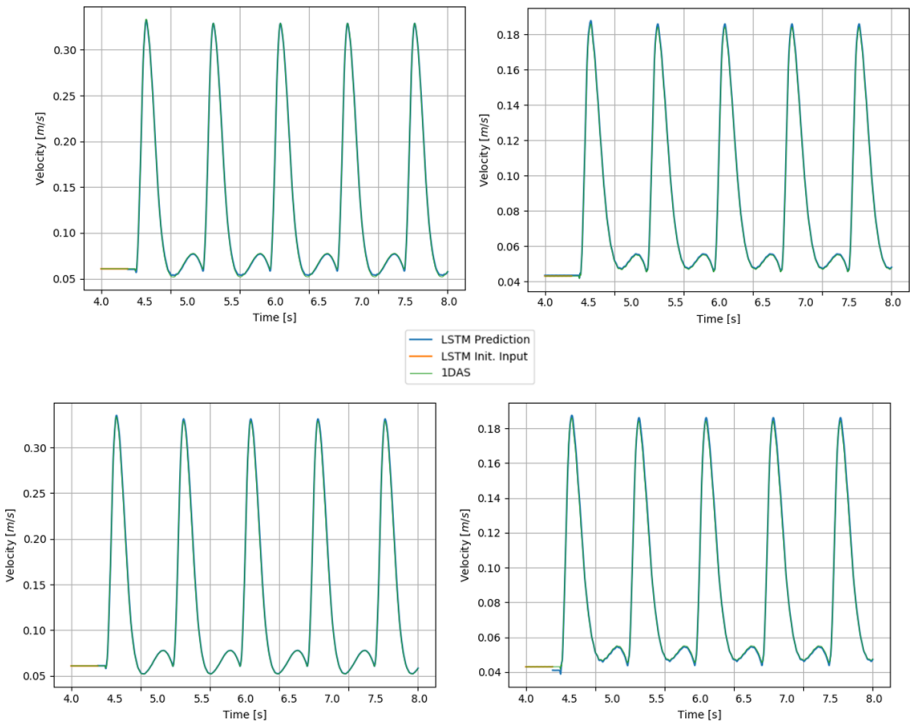
Σχήμα 6: *Η πραγματική παροχή αίματος (πάνω) και οι χρονοσειρές/προβλέψεις της ταχύτητας στον 50ο κόμβο, δείγματα τα οποία δεν έχει δει το δίκτυο (κάτω).*

Μεταβαλλόμενο Πάχος Τοιχώματος Αρτηρίας

Η κατανομή του πάχους των τοιχωμάτων είναι μεταβαλλόμενη κατά μήκος της αρτηρίας. Αυτό παραμετροποιείται με μία καμπύλη Bezier με 7 ΣΕ, όπως και το αρχικό σχήμα της αρτηρίας. Μεταβάλλοντας τα ΣΕ των καμπυλών οι οποίες παραμετροποιούν το αρχικό σχήμα και το πάχος, το 1DAS παράγει κατανομές ταχύτητας. Εκπαιδεύονται δύο δίκτυα, με τον ίδιο τρόπο. Το πρώτο έχει ως είσοδο τις 150 προηγούμενες κατανομές και τις 2 παραμετροποιημένες κατανομές. Το άλλο έχει μόνο τις 150 κατανομές. Τα αποτελέσματα και των δύο είναι πολύ ικανοποιητικά, με το δεύτερο να παρουσιάζει λίγο καλύτερα αποτελέσματα, Σχ. 7, με μέσο σφάλμα MAE $1.2 \times 10^{-3} m/s$.

Βελτιστοποίηση Εισόδου και Αρχιτεκτονικής του Δικτύου

Χρησιμοποιείται η στατιστική μέθοδος Μερικής Αυτόματης Συσχέτισης, με την οποία υπολογίζεται από πόσες προηγούμενες κατανομές εξαρτάται η παρούσα και κατά συνέπεια πόσες κατανομές πρέπει να είναι είσοδος στο δίκτυο. Το αποτέλεσμα είναι 60 κατανομές ταχύτητας.



Σχήμα 7: *Προβλέψεις από το δίκτυο το οποίο τροφοδοτείται και με τις κατανομές εμβαδού και πάχους (πάνω) και προβλέψεις από το δίκτυο το οποίο δεν τροφοδοτείται από αυτές (κάτω). Όλα τα διαγράμματα παρουσιάζονται στο 50ο κόμβο.*

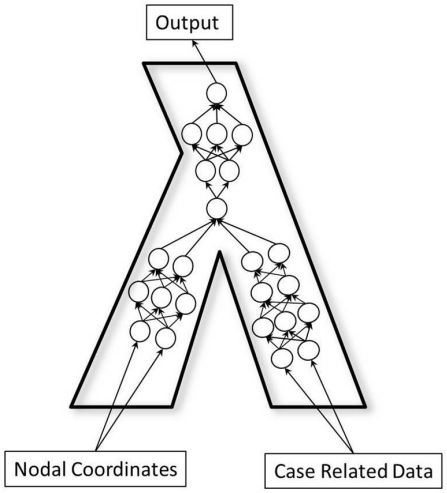
Επίσης με τη χρήση ενός αλγόριθμου ελαχιστοποίησης που χειρίζεται πληθυσμούς υποψήφιων λύσεων, γίνεται βελτιστοποίηση του αριθμού νευρώνων του δικτύου σε κάθε επίπεδο. Η αρχιτεκτονική που προκύπτει σε συνδυασμό με τον αριθμό των κατανομών μειώνουν το κόστος εκπαίδευσης σε $\sim 36mins$ με τα ίδια αποτελέσματα (μέσο σφάλμα MAE $1 \times 10^{-3} m/s$) με το προηγούμενο δίκτυο.

Πρόβλεψη Ροών με λ-DNN

Σε αυτό το κεφάλαιο, ένα άλλο είδος ΒΝΔ χρησιμοποιείται για την πρόβλεψη αεροδυναμικών ροών και κατανομών θερμοκρασίας. Αυτό το δίκτυο ανήκει στην κατηγορία των πολυκλαδικών δικτύων, με δύο κλάδους, όσα και τα είδη εισόδων σε αυτό. Η αρχιτεκτονική του παρουσιάζεται στο Σχ. 8.

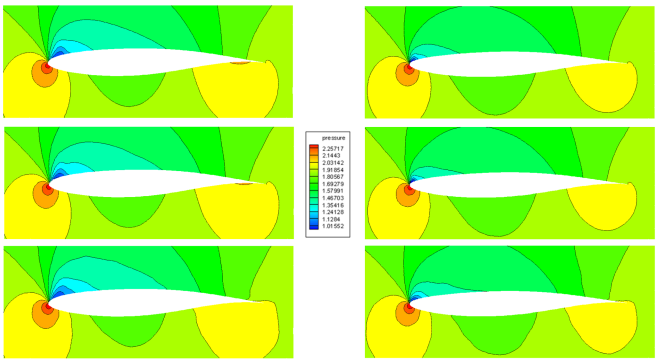
Πρόβλεψη Πεδίου Πίεσης γύρω από Αεροτομή

Το δίκτυο καλείται να προβλέψει 2Δ κατανομές πίεσης γύρω από μία αεροτομή. Η αεροτομή παραμετροποιείται με 2 καμπύλες Bezier, με 6 ΣΕ η καθεμία. Μεταβάλλοντας κάποια από αυτά προκύπτουν 180 αεροτομές και, κατά συνέπεια, 180 πεδία πίεσης. Το δίκτυο εκπαιδεύεται να προβλέπει την τιμή της πίεσης σε κάθε κόμβο τροφοδοτώντας το με τις συντεταγμένες του κόμβου και τις συντεταγμένες του περιγράμματος της αεροτομής.



Σχήμα 8: *Η αρχιτεκτονική του λ-DNN δικτύου.*

Τα αποτελέσματα του δικτύου συγκρίνονται με αυτά ενός κλασικού πλήρως διασυνδεδεμένου δικτύου (FCNN) με το λ-DNN να έχει σφάλμα MAPE 0.36% σε σχέση με το άλλο το οποίο έχει 0.50%, Σχ. 9.



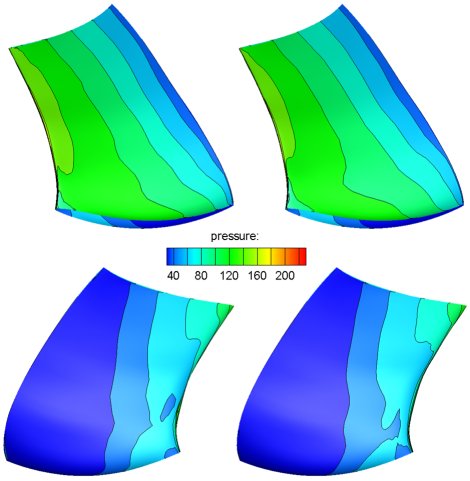
Σχήμα 9: *Προβλέψεις από το λ-DNN (μέση) και από το FCNN (κάτω) συγκρίνονται με τις προβλέψεις του λογισμικού ΥΡΔ (πάνω).*

Πρόβλεψη Πεδίου Πίεσης στην Επιφάνεια Δρομέα Francis

Στη δεύτερη εφαρμογή, το δίκτυο εκπαιδεύεται να προβλέπει κατανομές επιφανειακής πίεσης ενός δρομέα υδροστροβίλου Francis. Η γεωμετρία του δρομέα παραμετροποιείται με τη βοήθεια του λογισμικού GMTurbo και προκύπτουν 16 μεταβλητές σχεδιασμού. Οι είσοδοι του δικτύου είναι οι 3 συντεταγμένες το κόμβου για τον οποίο πραγματοποιείται η πρόβλεψη καθώς και οι 16 μεταβλητές σχεδιασμού. Η αρχιτεκτονική του δικτύου διαμορφώνεται ώστε να τροφοδοτηθεί με τις νέες εισόδους.

Το δίκτυο εκπαιδεύεται και επιτυχώς μπορεί να προβλέψει κατανομές επιφανειακής πίεσης στην επιφάνεια ενός δρομέα τον οποίο δεν έχει δει κατά την εκπαίδευση, Σχ.

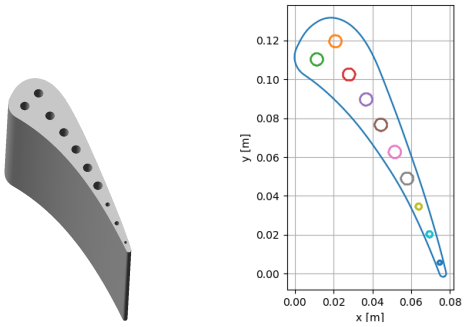
10, με σφάλμα MAPE 3%.



Σχήμα 10: *Επιφανειακές κατανομές πίεσης σε έναν δρομέα τον οποίο δεν έχει δει το δίκτυο από το λογισμικό ΥΡΔ (αριστερά) και από το λ-DNN (δεξιά).*

Πρόβλεψη Κατανομής Θερμοκρασίας σε Προσομοίωση Συζευγμένης Μεταφοράς Θερμότητας

Στην τελευταία εφαρμογή το δίκτυο χρησιμοποιείται σε ένα πολυπεδιακό πρόβλημα, στη Συζευγμένη Μεταφορά Θερμότητας (ΣΜΘ). Σε αυτή την ανάλυση ο επιλύτης του ρευστού επικοινωνεί ανταλλάσσοντας πληροφορία με τον επιλύτη της εξίσωσης αγωγής θερμότητας. Χρησιμοποιείται ένα εσωτερικά ψυχόμενο πτερύγιο θερμικής στροβιλομηχανής, με 10 κανάλια με σταθερές κυκλικές διατομές και σε σταθερές θέσεις, Σχ. 11 . Ο επιλύτης του ρευστού επιλύει τη ροή γύρω από το πτερύγιο και υπολογίζει τη θερμορροή στην περιφέρεια του πτερυγίου. Ο επιλύτης της εξίσωσης θερμικής αγωγής υπολογίζει την κατανομή της θερμοκρασίας λαμβάνοντας υπόψη την κατανομή της θερμορροής. Αυτή η επικοινωνία των δύο επιλυτών καθιστά την αριθμητική επίλυση του προβλήματος πολύ ακριβή.

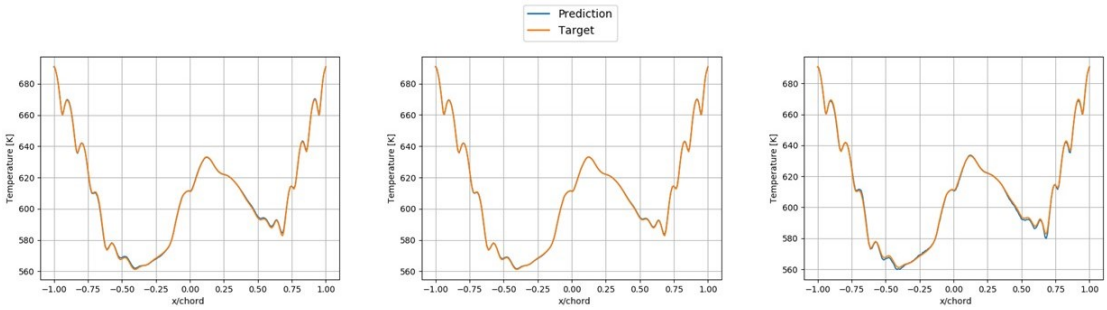


Σχήμα 11: *Το πτερύγιο σε 3Δ προοπτική (αριστερά) και σε διάγραμμα (δεξιά).*

Το λ-DNN δίκτυο εκπαιδεύεται ώστε να υποκαθιστά τον επιλύτη της εξίσωσης αγωγής της θερμότητας και έτσι έχει ως εισόδους την κατανομή της θερμορροής στο περίγραμμα

του πτερυγίου και τις συντεταγμένες της περιφέρειας. Η έξοδος αυτού ήταν οι θερμοκρασίες στην περιφέρεια (σε αντίθεση με τις προηγούμενες εφαρμογές το δίκτυο εδώ προβλέπει απευθείας την κατανομή και όχι από κόμβο σε κόμβο).

Η αρχιτεκτονική του δικτύου αλλάζει σε σχέση με τις προηγούμενες εφαρμογές. Τα αποτελέσματα του δικτύου είναι πολύ ικανοποιητικά, Σχ. 12, με σφάλμα MAPE 0.18%. Αυτά συγκρίνονται με ένα FCNN δίκτυο, με σφάλμα MAPE 1.52%, καθώς και με την αρχιτεκτονική του λ-DNN των προηγούμενων εφαρμογών, με MAPE 0.30%. 2 βελτιστοποιήσεις βασιζόμενες σε εξελικτικό αλγόριθμο επιβεβαιώνουν τις δυνατότητες του νέου δικτύου, και έχουν ως στόχο την ελαχιστοποίηση των απωλειών ολικής πίεσης μεταξύ εισόδου και εξόδου του ρευστού και της θερμοκρασίας στο πτερύγιο. Στην πρώτη βελτιστοποίηση, το χρησιμοποιείται λ-DNN, ενώ η δεύτερη βασίζεται αποκλειστικά στο λογισμικό της προσομοίωσης ΣΜΘ. Τα μέτωπο μη κυριαρχούμενων λύσεων της πρώτης βελτιστοποίησης κυριαρχεί έναντι του άλλου.



Σχήμα 12: *Η κατανομή θερμοκρασίας (πρόβλεψη) ενός πτερυγίου, το οποίο δεν έχουν δει τα δίκτυα. Τα αποτελέσματα από το λ-DNN (προηγούμενες εφαρμογές) (αριστερά), από το τροποποιημένο λ-DNN (μέση) και το FCNN (δεξιά).*

Συμπεράσματα

Και τα δύο είδη δικτύων είναι ικανά να προβλέψουν πεδία ροών. Το δίκτυο LSTM μπορεί να προβλέψει χρονικά μεταβαλλόμενα μεγέθη. Μπορεί να ανατροφοδοτεί τις προβλέψεις στον εαυτό και να τις χρησιμοποιεί για μελλοντικές και έτσι απαιτεί πολύ λίγη πληροφορία για την πρόβλεψη χρονοσειρών. Επίσης είναι ικανό να προβλέπει χρονοσειρές τροφοδοτώντας το μόνο με προηγούμενες κατανομές, χωρίς να έχει πληροφορία για την παραμετροποίηση. Το λ-DNN μπορεί να προβλέψει αεροδυναμικές ροές με μεγάλη ακρίβεια καθώς και να χρησιμοποιηθεί σε πολυπεδιακά προβλήματα, αντικαθιστώντας τον επιλύτη του ενός πεδίου. Μάλιστα, οι προβλέψεις του μπορούν να αξιοποιηθούν στο τομέα της βελτιστοποίησης.