

École polytechnique de Louvain

Supervised Learning Tools applied to Aircraft Wake Detection and Formation Flight

Author: **Grégoire VEKEMANS**
Supervisor: **Pr. Philippe CHATELAIN**
Teaching Assistant: **Ignace RANSQUIN**
Readers: **Pr. Christophe DE VLEESCHOWER, Pr. Grégoire WINCKELMANS**
Academic year 2020 - 2021
Master [120] in Mechanical Engineering

Everybody has already seen some geese flying in close V-formation. But did you know that they are actually saving energy which allows them to fly across much more long distances ? Although this technique improves a lot the performances of aircraft, it remains very dangerous because of the hard downwash at the center of the wake. Therefore the safety and the control of wakes are widely studied which comes up to the question of observability. How is a wake generated and how does it work ? Is a wake always observable ? Which onboard measures are useful in this task ?

Wakes may be characterized as two big vortices generated at the wingtips of an aircraft. The wake thus generated will modify the wind speeds in the environment which will affect any aircraft encountering it. It will actually have an influence on its aerodynamic efforts which will give valuable information. However their complexity and the high non-linearity of the system avoids any analytical solution to be found. The path of supervised learning tools are then studied. In that respect, we will use neural networks and observe their performances for observing a wake.

Supervised learning tools' ability to fit any function is well known and the wake of an aircraft makes no exception. They show that they are able to understand the high non-linearity of aerodynamical efforts induced by the wake of an aircraft. Besides, there remain some areas where they lack in performances but these are finally understood as being zones of unobservability of the system.

ACKNOWLEDGEMENT

I would like to thank the supervising team, namely Philippe CHATELAIN and Ignace RANSQUIN, in charge of this project without who it could not happen. Their weekly support and advices helped me all along to make short term decisions without forgetting my objectives. They did neither lack in boosting me when results did not show up or when bugs were following on.

I would also like to thank my housemate Jean-Baptiste CRISMER with who I had a lot of discussion about his thesis and mine. The chance that we had to live together while having the same promotor pushed me forward and helped me a lot to keep focus.

Next in the list is Kylian TAMINIAU with his helpful critical eye. His misunderstanding of the subject helped me a lot to verbalize it and deep my own understanding of it. His weekly interest also helped me not to give up and to throw all away when problems came up to persist for too long.

Last but not least, I would like to thank a lot all my family and my parents especially. I could not achieve anything if it was not them. They as well as Thibault HERARD and Pierre-Louis HANCE showed great skills when times came to read my work and enlighten language errors and unclear paragraphs.

To all other people who heard me talk about that crazy project of flying in formation to save only 5 to 10% of fuel, I might overwhelmed things and you surely did not understand a word of what I said. However, thank you a lot for your time and interest.

1	Introduction	1
2	Aircraft Dynamics	3
2.1	System of Axes	3
2.2	Aerodynamic Forces	4
2.3	Prandtl Lifting Line Theory	5
2.4	Aircraft Control	6
2.5	Numerical Tool	9
3	Supervised Learning	11
3.1	Neural Networks	12
3.2	Feed-Forward Neural Networks	15
3.3	Recurrent Neural Networks	16
3.4	Building a Model	17
3.5	Training and Validation	20
3.6	Testing	21
4	Learning Models	23
4.1	Database	23
4.2	Specifications	25
4.3	Expectations	29
5	Performances	31
5.1	Side Prediction	31
5.2	Orientation Approximation	36
5.3	Approach Prediction	38
5.4	Discussion	41
6	Conclusion	45
A	Animation of a Random Target Policy	47
B	Lifting Line Theory	53
C	Training Performances	55
D	Animation of the Wake's Observation	57

Abbreviations

AC	Aircraft
CCE	Categorical Cross-Entropy
GRU	Gated Recurrent Unit
HT	Horizontal Tail
LL	Lifting Line
LSTM	Long-Short Term Memory
MAE	Mean Absolute Error
MSE	Mean Squared Error
NN	Neural Network
RNN	Recurrent Neural Network
VT	Vertical Tail

Aircraft Dynamics

α	Angle of attack or incidence perturbation
β	Sideslip angle perturbation
$\ddot{x}$	Longitudinal acceleration in the earth-fixed frame
$\ddot{y}$	Lateral acceleration in the earth-fixed frame
$\ddot{z}$	Normal acceleration in the earth-fixed frame
Γ_0	Intensity of wingtip vortices
ϕ	Roll angle
ψ	Yaw angle
σ	Standard deviation
θ	Pitch angle

ε	Error on the approximation of the orientation
b	Wingspan
b_0	Distance between the centers of wingtip vortices
m	Aircraft mass
u	Longitudinal velocity in the earth-fixed frame
v	Lateral velocity in the earth-fixed frame
V_∞	Tangential freestream velocity
w	Normal velocity in the earth-fixed frame
x	Longitudinal coordinate in the earth-fixed frame
y	Lateral coordinate in the earth-fixed frame
y_L	Lateral coordinate of a leader aircraft in the earth-fixed frame
z	Normal coordinate in the earth-fixed frame
z_L	Normal coordinate of a leader aircraft in the earth-fixed frame

Whenever a plane generates lift, it also generates induced drag which is an undesired yet unavoidable result of the pressure difference between the upper and lower surfaces of the wing. More than that, the wake created at the back of the aircraft could also cause a potential threat and harm another aircraft which encounters it. Indeed, local air speeds near the wake could be very large and could induce strong reactions for the encountering aircraft. This sometimes lead to passenger injuries, loss of control or even crash and death. This is the reason why there are so much rules for regulating the air traffic and keep a minimal distance between two aircraft ^[1,2].

This being said, the wake of an aircraft is characterized as two large vortices resulting from the roll up of the vortex sheet emanating from the main lifting surfaces of the aircraft. Although they represent a non-negligeable threat, they could be used in a beneficial way. Because of the their rotational direction, the two vortices push the air downward between them, while they are pushing it upward in the outer regions. Therefore, any aircraft flying in one of these can take advantage of the upwash induced speeds. The energy that it receives from the wake then allows it to save fuel as well as reduce its drag! This method of energy saving has been already seen through the typical V-formation flight of birds such as migrating geese ^[3,4]. Formation flight has then been applied to fixed-wing aircraft ^[5,6].

While birds are known for flying in close formation, it is more common for civil aviation to apply an extended version for obvious safety reasons. Besides, formation flight application has been first applied to battle strategy and assistance so it is not surprising for us to learn that battle aircraft already use formation flight quite efficiently, i.e. the U.S. Air force developped a flight controller for their C-17 bombarders which purpose is the maintaining of a relative position – called sweet spot – between the aircrafts flying in formation ^[7]. The past decade, Boeing and Airbus yet showed their interest in the integration of formation flight into civil aviation ^[8,9]. Studies actually conducted some tests in close formation flight and confirmed a fuel reduction from 5 to 10% at small lateral distances.

The research about formation flight conducted to two distinct strategies for tracking and sensing the wake of a leader aircraft:

- The first one consists in measuring the spanwise distribution of circulation, which is equivalent to measuring the pressure difference. Although this strategy is based on the fundamentals of aircraft dynamics and showed accurate results, the need of sensors aboard the follower aircraft imposes new constraints on wing manufacturing which may become expensive.
- The second strategy is conversely based on the measurements of sensors – flight dynamics and controls – with which current aircraft are already equipped. Hence, these measurements being non-linear functions of the circulation, it results in a more challenging problem.

This second strategy has been adopted. However, the high non-linearity of measures considerably increases the complexity of the problem. Therefore it is impossible to characterize the wake analytically, based on these onboard measurements. Hopefully, supervised learning tools have shown good performances in fitting any curve

depending on the complexity of the selected algorithm. This is the reason why machine learning tools – and more specifically neural networks' algorithms – are considered for the tracking.

In this work, we will first go through the fundamentals of aircraft dynamics and control, and we will talk about the wake formation and the numerical tool in Chapter 2. Then, the neural networks algorithms, ability and hyperparameters are presented in Chapter 3. Chapter 4 is about the building of the database and the specifications of inputs and outputs of studied learning models. Finally, we will present training and testing performances of chosen neural networks' architecture followed by a discussion in Chapter 5.

According to the Cambridge Dictionary the dynamic of an object is *the study of forces and torques acting on that object, resulting in a production of movement*. Theories about aerodynamics and flight dynamics have been widely investigated^[10,11] and this work does not aim to go against them; yet this chapter gives fundamentals for the reader to understand the further work.

First, we will sketch a set of usefull systems of axes. Then we will go through the definitions of aerodynamic forces, vortex sheets and some dynamic controls for a better understanding of the numerical tool developed for a simplified aircraft.

2.1 System of Axes

Since an aircraft has six degrees of freedom, the description of its motion can be relatively complex. Therefore, motion is usually described by a number of variables which are related to a suitably chosen system of axes. For the sake of simplicity we use axes and notations that M. Cook used in his book^[10]; they are presented in the few next paragraphs and plotted on Fig. 2.1.

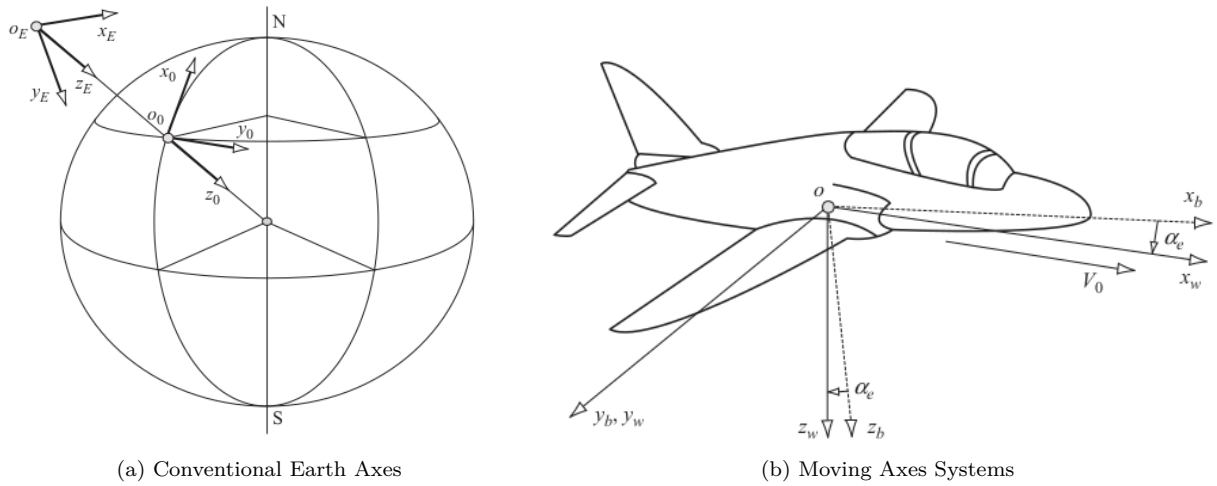
First, only normal atmospheric flight is considered such that it is usual to measure aircraft motion with reference to an *earth-fixed frame*. A reference point o_0 – determined on the surface of the earth – is the origin of a right-handed orthogonal system of axes $(o_0x_0y_0z_0)$, where o_0x_0 points to the north, o_0y_0 points to the east, and o_0z_0 points vertically “down” along the gravity vector. A motion may be completely described by such a model, although it is actually best suited for navigation and performance applications where flight path trajectories are of primary interest.

For flight dynamics applications, we rather use a simplified definition of earth axes; since only short-term motion is of interest, it is perfectly adequate to assume that the flight occurs above a flat earth¹. In that respect, the aircraft would stay at a constant altitude, such that the plane $(o_Ex_Ey_E)$ is parallel to the plane $(o_0x_0y_0)$ at the surface of the earth. Though pay attention to the fact that the axis o_Ex_E does not point to the north anymore but in the direction of the flight.

In addition to the earth-fixed framework, a number of aircraft body-fixed axis systems are in common use too, namely a generalised one and the wind – or aerodynamic – axes. These are mainly used for small perturbation analysis. It is actually usual practice to define a right-handed orthogonal axis system fixed in the aircraft and constrained to move with it. The most general one known as a *body axis* system $(ox_by_bz_b)$ is defined as (ox_bz_b) being the plane of symmetry of the aircraft, and the ox_b axis being parallel to the geometrical *horizontal fuselage datum*. In that respect, the oy_b axis is directed to starboard and the oz_b is directed downward.

More specifically, the body axes system is a generalisation of earth-fixed axes system for which the angular states of the aircraft are considered. In that respect, the earth-fixed frame is first moved of ψ angle around gravity

¹Dear Galileo, please do not mind that and rest in peace. See you in heaven!

Figure 2.1: Axes Systems^[10]

vector, then of θ pitch angle such that the nose points into the right direction, and finally of ϕ roll angle around the ox_b axis. Mathematically speaking, the new axes system reads

$$\begin{pmatrix} ox_b \\ oy_b \\ oz_b \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{pmatrix} \cdot \begin{pmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{pmatrix} \cdot \begin{pmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} ox_0 \\ oy_0 \\ oz_0 \end{pmatrix} \quad (2.1)$$

On the other part, the *wind* axes are defined such that the ox axis is parallel to the velocity vector $\mathbf{V}$. It is supported by the fact that a plane is not always flying in the direction of its nose, which means that the velocity vector is not parallel to the ox_b axis from the body axes system. In steady symmetric flight, wind axes ($ox_w y_w z_w$) are just a particular version of the body axes which are rotated around the oy_b axis through the steady body angle of incidence α – also called angle of attack – until the ox_w axis aligns with the velocity vector. However the flight may not be symmetric anymore, when the body axes system is rotated around the oz_b axis through the angle of slip β . As an example, landing aircraft are often forced to land while their nose is not pointing in the direction of the airstrip. These two angles are important information for the pilot because they help him to know how his aircraft is placed with respect to the direction it is flying to.

2.2 Aerodynamic Forces

The total aerodynamic force acting on a body is usually thought of as having two components; the first one is parallel to the incoming flow and is called the *drag* while the second one is perpendicular and is called the *lift*. The object is generally thought such that at practical angle of attack, the lift greatly exceeds the drag^[12,13]. When a flow is coming up to a wing, it changes its direction which produces lift. This change of acceleration and/or amplitude implies acceleration, which means a force. Therefore, changing the direction of the flow requires a force to be applied to the fluid; the reaction force of the fluid acting on the wing is the lift.

The induced lift and drag are schemed for a 2d-airfoil on Fig. 2.2. The induced drag depends on the amount of downwash the wing is subjected to, which depends itself on the angle of attack of the wing. We observe that the lift is perpendicular to the freestream velocity, and spaced of the same downwash angle from the effective lift $\mathbf{L}_{\text{eff}}$, which is the reaction force of the fluid acting on the wing. That force is perpendicular to the true air speed as felt by the wing. The component of $\mathbf{L}_{\text{eff}}$ parallel to the incoming flow is the drag induced by the flow on the wing.

In other words, the change of direction induces a pressure distribution along the wing. Since the air below the wing is at higher pressure than the air above, it produces the lift. On a wing of finite span, this pressure difference causes air to flow from the lower surface, around the wing tip, towards the upper surface. The combination of this spanwise flow with the chordwise flowing air causes a change in speed and direction, which finally results in

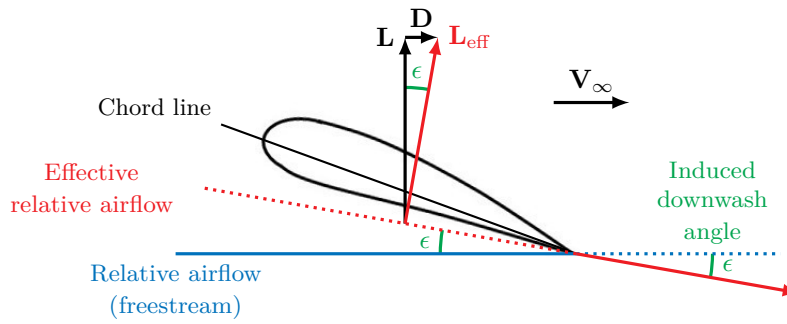


Figure 2.2: Airfoil Induced Lift and Drag

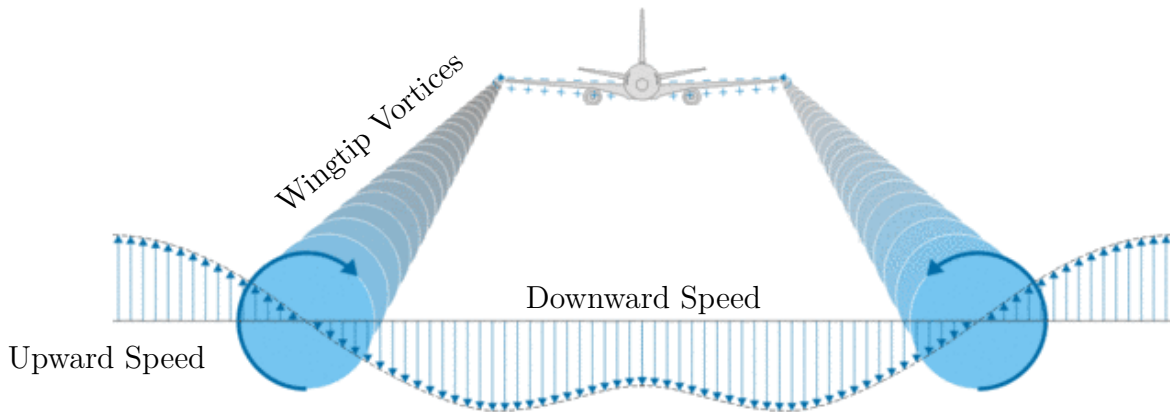


Figure 2.3: Rear View of a Flying Aircraft

a twisting of the airflow and a production of vortices along the wing trailing edge. These vortices are unstable, which is why they quickly combine and produce wingtip vortices^[14]. Then it propagates behind the trailing edge, deflecting it downwards, and thus inducing downwash behind the wing.

2.3 Prandtl Lifting Line Theory

Just as stated before, wingtip vortices are the signature of a wing generating lift. They are characteristic through their circular pattern of rotating air, as schemed on Fig. 2.3. The name wingtip vortices may be slightly misleading because the air twisting occurs all along the wingspan. Vorticity is effectively trailed on every points of the wing and eventually rolls up into larger vortices at the wingtip^[14,15].

Those wingtip vortices are the main characteristic of an aircraft wake. They are schemed on Fig. 2.3. We observe there the two infinite length vortices on both sides of the aircraft. On the front of the figure is presented the induced air speed; just at the back of the aircraft, these speed are downward while on the exterior parts they are upward. Therefore, one deduces that the behavior of the wake is symmetrical: indeed, whereas the right vortex is rolling in the trigonometric direction, the left one is rolling in the other direction. These two vortices have the same intensity everytime and the distance to the center of the wake is the same. Therefore what happens at starboard to the center of the wake is similar to what happens portside and studying the wake on a single side is not meaningless.

All of this being said, how can we use it for describing an aircraft's motion ? Since all the points of the wings are playing a role in the formation of wingtip vortices, scientifics are commonly referring to the phenomenon as the *vortex sheet*. This concept has been described by the *Prandtl Lifting Line theory* which is the matter of the next few paragraphs. The theory has been developped by Ludwig Prandtl in 1918-1919 who aimed to build a mathematical model that predicts the lift distribution over a thin 3D wing with a high aspect ratio^[11,16].

First of all, the model starts from a simplified 2d-view study of the wing just as we talked about in Section 2.2. This first approximation considers the wing into a set of cross-sections – airfoils – which are easier to understand than a complete wing. However, the behavior of the wing does not correspond to the sum of all independtly-computed airfoils as one might expect. Indeed, the air twisting at the surface of the wing involves a 3d-analysis that a 2d study cannot predict. The lifting line theory therefore corrects some of these errors by including some of the interactions between wing cross-sections. It then produces the lift distribution $l(y)\mathbf{e}_z$ along the span-wise direction and the flow conditions (ρ, V_∞, α) .

The LL theory applies the concept of circulation via the Kutta-Jukowski theorem which gives the sectional lift density as

$$l(y) = \rho V_\infty \Gamma(y). \quad (2.2)$$

The last term of the right-hand member is what we called the *circulation* and is defined as the curl integral of the flow velocity:

$$\Gamma \triangleq \oint_C \mathbf{V} \cdot d\mathbf{l}. \quad (2.3)$$

Through Eq. (2.2), the unknown is not the lift distribution anymore but the circulation $\Gamma(y)$ of air along the span. It gives the advantage of taking into account the mutual influence of cross-sections between each other. Therefore, any change of the circulation over the spanwise will induce an equivalent change in lift. Moreover H. Helmholtz stated that vorticity cannot begin or terminate in the air: it means that the dissipation of a vortex filament down the flow can model any spanwise change in lift^[17]. This shed vortex, whose strength is the derivative of the local wing circulation distribution, $\frac{d\Gamma}{dy}$, influences the flow left and right of the wing section.

As mentionned above, the lateral influence is the key to LL theory. If the change in lift distribution is known at a given lift section, it is possible to predict what is the influence in lift of that section over its neighbors: the vertical induced velocity can be quantified using the velocity distribution within a vortex, and related to a change in effective angle of attack over neighboring sections.

In mathematical terms, the local induced change of angle of attack α on a given section is quantified with the integral sum of downwash induced by every other wing section. In turn, the integral sum of sectional lift density on each downwashed wing sections is equal to the total desired amount of lift, and the induced drag follows equivalently. These finally read

$$L = \rho V_\infty \int_{-b/2}^{b/2} \Gamma(y) dy \quad (2.4)$$

$$D_i = \rho V_\infty \int_{-b/2}^{b/2} \epsilon(y) \Gamma(y) dy \quad (2.5)$$

where $\Gamma(y)$ is expressed in terms of the wing geometry – aspect ratio, chord length and local angle of attack of local cross-sections etc. – and its own derivative $\frac{d\Gamma(y)}{dy}$. This expression is developped in Appendix B for the simple case of untwisted and thin airfoil. Moreover, the direction of freestream velocity is taken to be perpendicular to the spanwise which is the most common case but does not consider any slip angle.

Nevertheless, it seems necessary to keep in mind that the LL theory of Prandtl stays a rough approximation. Although it assumed at first that wings are rigid, a recent work has improved the model by adding mollification characteristics^[18]. In addition to that, aircrafts are flying at subsonic speeds, which means that air could be assumed to be incompressible and inviscid. Some other characteristics of airfoil are not taken into account – such as the flaps, the cambered, etc. – as well as the aircraft fuselage is not considered. In aim to describe the aircraft more accurately, one may nonetheless describe it as being a set of three LLs, the main one being the wing, the two others being the vertical and the horizontal tails.

2.4 Aircraft Control

Since forces and moments describe the interaction of an aircraft with its environment and that we can derive from them its linear and angular velocities, its behavior might be completely resolved into components with

X	Axial drag force	Sum of the components of aerodynamic, thrust and weight forces
Y	Side force	
Z	Normal lift force	
L	Rolling moment	Sum of the components of aerodynamics, thrust and weight moments
M	Pitching moment	
N	Yawing moment	
p	Roll rate	Components of angular velocity
q	Pitch rate	
r	Yaw rate	
U	Axial velocity	Total linear velocity components of the center of gravity
V	Lateral velocity	
W	Normal velocity	

Table 2.1: Perturbation Variables^[10]

respect to the chosen aircraft fixed axis system. Therefore it is convenient to use a generalized body axis system, initially assuming a steady straight flight with a constant body incidence α_e and a steady velocity V_∞ which resolves into components U_e , V_e and W_e . Of course when the aircraft is at trimmed equilibrium, the flight is steadily nonaccelerating (constant velocities)^[10].

Whenever the aircraft is disturbed from equilibrium, the resulting motion is quantified in terms of the perturbation variables shown on Fig. 2.4. Then, a simple description of the latter is given in Table 2.1. Let's keep in mind that the components of the total linear velocity perturbations (U, V, W) are given by the sum of the steady equilibrium components and the transient perturbation components (u, v, w), this means

$$\mathbf{V} = \begin{pmatrix} U \\ V \\ W \end{pmatrix} = \begin{pmatrix} U_e + u \\ V_e + v \\ W_e + w \end{pmatrix} \quad (2.6)$$

Because of the system of axes defined previously, one easily deduces that $U_e = W_e = 0$. Therefore the previous equation simplifies as

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \begin{pmatrix} u \\ V_\infty + v \\ w \end{pmatrix} \quad (2.7)$$

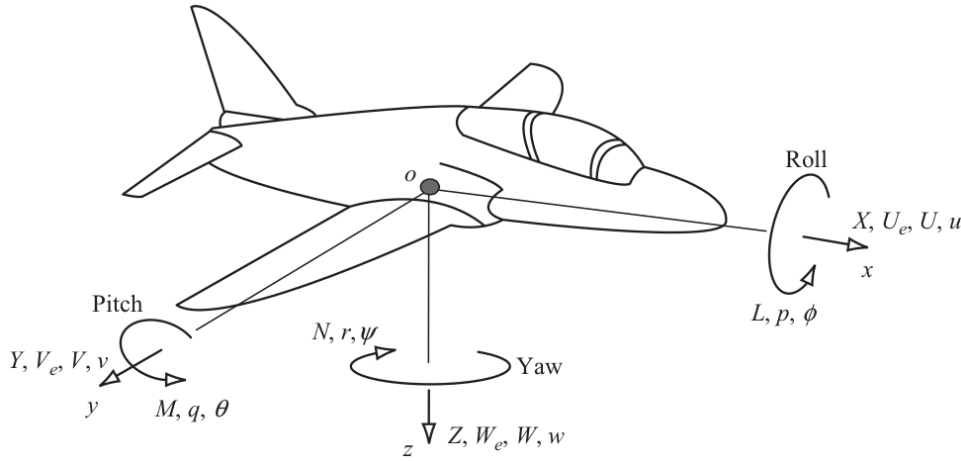
While an aircraft is disturbed from its trimmed equilibrium, it has four commands for control. It allows it to steer into the right direction. Aerodynamic effects and related commands are described hereunder:

Thrust Force that pushes the aircraft forwards. It is created through the motors and steer in the direction of the nose. There are several types of propulsion device – such as propellers with an internal combustion engine, turbojets, turbofans and more recently turbopropeller – which determines the type of flight.

Roll Wing control that allows the aircraft to rotate around the ox axis. They are placed near the wingtips for two reasons: the center of mass of the aircraft is at the center of the principal wing, and the moment is dependent to the distance with the center of mass. In terms of displacement, actionning this command moves the aircraft to the left or the right. Even if the plane is moving on one side, its nose always steers into the same direction. In other words, the aircraft will move along the oy axis.

Pitch Tail control that allows the aircraft to rotate around the oy axis. The aircraft point its nose whether to the sky or to the earth. By efficiently controlling the thrust, the aircraft will be able to gain or lose altitude, respectively.

Yaw Tail control that allows the aircraft to rotate around the oz axis. The aircraft is now steering wheter to the left or to the right. However, an aircraft having an initial velocity along the ox axis and pointing its nose along the oy axis will keep its initial velocity while accelerating in the new direction it is steering to. In aim to turn efficiently, we must then adjust both the roll and the yaw such that the aircraft is moving in the direction its nose is steering.

Figure 2.4: Motion Variables Notation ^[10]

In Table 2.1, the rotation rates p, q, r are the angular velocities expressed in the earth-axes frame. Since body-fixed variables ϕ, θ, ψ are actually angles which aim to switch of coordinate system of axes, one cannot take their temporal derivation as being the roll rates. In fact, these are defined as

$$\begin{pmatrix} p \\ q \\ r \end{pmatrix} = \begin{pmatrix} \dot{\phi} - \dot{\psi} \sin \theta \\ \dot{\theta} \cos \phi + \dot{\psi} \sin \phi \cos \theta \\ \dot{\psi} \cos \phi \cos \theta - \dot{\theta} \sin \phi \end{pmatrix} \quad (2.8)$$

However, because of the assumption that ϕ, θ, ψ are small angles, these relations are simplifying and we finally assume for now on that

$$\begin{pmatrix} p \\ q \\ r \end{pmatrix} \approx \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} \quad (2.9)$$

The main challenge of controlling an aircraft is thus the ability of keeping it stable while it is subjected to perturbations. One is able to define optimal commands while solving the equation of motion for the aircraft which are derived from Newton's laws of motion. These law are built following

$$\begin{aligned} \text{Force} &= \text{Mass} \times \text{Inertial Acceleration} \\ \text{Moment} &= \text{Moment of Inertia} \times \text{Angular Acceleration} \end{aligned} \quad (2.10)$$

By considering a rigid body motion with 6 degrees of freedom, one then develops the 6 equations of motion. The first three of them concerns the linear forces in the three axes directions. The three others concern the angular momentum around the three axes. These six equations are vectorized as

$$\begin{aligned} \mathbf{F} &= m (\dot{\mathbf{V}} + \boldsymbol{\omega} \times \mathbf{V}) \\ \mathbf{M} &= m [\mathbf{r} \times (\dot{\mathbf{V}} + \boldsymbol{\omega} \times \mathbf{V})] \end{aligned} \quad (2.11)$$

where $\mathbf{V}$ is the total velocity of the center of gravity, $\boldsymbol{\omega}$ is the angular velocity as stated by Table 2.1 and $\mathbf{r}$ is the positional vector. The vectors $\mathbf{F}$ and $\mathbf{M}$ represent the linear forces and the angular momenta, respectively. One simplifies their understanding by considering them as being due to aerodynamic effects, gravitational effects, movement of aerodynamic controls, power effects, and effects of atmospheric disturbances.

Such a system may become very spurious and bring much difficulties when solving because of the great amount of elements it depends on. This is the reason why it is widely accepted to split it into two main parts which correspond to the longitudinal and the lateral-direction equations of motion. The first one will handle forces and torques that mainly act in the oxz plane, it means the axial force X , the normal force Z and the pitching moment M . On the other part, lateral-direction equations of motion will handle the side force Y , the rolling

moment L and the yawing moment N . Of course, all forces and torques have an impact on each other such that both simplified dynamics depend on the other one. We finally may determine some control through a linearized system of previous equations.

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)\end{aligned}\tag{2.12}$$

where

- $\mathbf{x}(t)$ is the column vector of n state variables, called the *state vector*,
- $\mathbf{u}(t)$ is the column vector of m input variables, called the *input vector*,
- $\mathbf{A}$ is the $(n \times n)$ *state matrix*,
- $\mathbf{B}$ is the $(n \times m)$ *input matrix*,
- $\mathbf{y}(t)$ is the column vector of r output variables, called the *output vector*,
- $\mathbf{C}$ is the $(r \times n)$ *output matrix* and
- $\mathbf{D}$ is the $(r \times m)$ *direct matrix*.

Since the aircraft control system is highly non-linear, it is not possible to find a solution without having linearized it. In that respect, it is common to choose the trim equilibrium flight as reference point. But it means that if we go far from it, the probability for the control to become unstable increases. The controller is thus optimized for small perturbations and displacements.

2.5 Numerical Tool

Although an aircraft model built on LLs theory with a linearised control system based on its 6 equations of motion is performant, it still remains a simplification of what happens for real. Hopefully, there exists another way for computing an aircraft model which consists in using numerical methods. Yet, it has the advantage of being much more accurate, its drawback is the computation costs. Since it needs much more computational resources, the simplified model is preferred. Nevertheless, in aim to increase its performances, some validations has been handled such that results obtained from both the numerical methods and the LLs theory coincide.

The model has been recently developped at UCLouvain by Ignace RANSQUIN in the `Python` programming language^[19]; it computes aerodynamic efforts that an aircraft is subjected to and integrates them over time in aim to deduce its motion. At each time step, we are therefore able to know the exact state of the aircraft as well as several useful measures which are presented in Table 2.2. The state of the aircraft is represented by all measures used for computing both the aerodynamic efforts and the control commands. These are of course available in measures' vector.

In addition to that, the source code also proposes a set of possibilities which allow the user of making his simulation unique. It involves a target policy and a set of external aircrafts; Since equations of motion are solved, it is possible to give a target on which we want the aircraft to go. This might be whether a position or a velocity target at user's choice. However, it is difficult to control the velocity of the aircraft because it can fly far away before we can notice it. This is the reason why we will prefer to use the positional target. Note that it will transpose the position target into a velocity target such that the aircraft splits its displacement into several steps. The choice has also been made to keep a constant target in the ox direction because high displacements along this has revealed to be unstable.

Finally, it is also possible for the user to modify the aircraft environment by initiating a set of external aircraft. It includes their linear positions and their roll (x_L , y_L , z_L and ϕ_L), their linear velocities ($\dot{x}_L$, $\dot{y}_L$ and $\dot{z}_L$) and the characteristics of their vortex sheet ($\Gamma_{L,0}$ and $b_{L,0}$). As it was mentionned before, their passage will sign a wake in the environment from which will results two big vortices. These vortices will then induce air velocities which modify the aerodynamic efforts of the aircraft. Because these induced efforts at the center of the wake make the aircraft's control unstable, we avoid the latter of going to close.

This numerical tool for computing the aircraft dynamic is efficient if the aircraft is well configured. In fact, the dynamic is highly dependent to the aircraft's dimensions and controller parameters. Since the source code is quite new, we were able to work with a single type of aircraft which is the c-172. A flight path of a simulated aircraft for the same target policy as in Section 4.1 is presented in Appendix A. These figures are splitted into

Body frame	Earth frame	Description
$\dot{x}_l$	$\dot{x}$	Axial velocity
$\dot{y}_l$	$\dot{y}$	Lateral velocity
$\dot{z}_l$	$\dot{z}$	Normal velocity
$\dot{\phi}$	p	Roll rate
$\dot{\theta}$	q	Pitch rate
$\dot{\psi}$	r	Yaw rate
x_l	x	Axial position
y_l	y	Lateral position
z_l	z	Normal position
ϕ		Roll
θ		Pitch
ψ		Yaw
α		Angle of attack
β		Angle of slip
γ		Flight path angle perturbation
	$\ddot{x}$	Axial acceleration
	$\ddot{y}$	Lateral acceleration
	$\ddot{z}$	Normal acceleration
$\ddot{\phi}$	$\dot{p}$	Roll acceleration
$\ddot{\theta}$	$\dot{q}$	Pitch acceleration
$\ddot{\psi}$	$\dot{r}$	Yaw acceleration

Others	Description
δ_ξ	Roll control stick angle
δ_η	Pitch control stick angle
δ_ζ	Rudder pedal control angle
τ	Engine thrust
Γ_0	Intensity of wingtip vortices
b_0	Distance between the centers of wingtip vortices
Δx	Difference in position with the target along the ox axis
Δy	Difference in position with the target along the oy axis
Δz	Difference in position with the target along the oz axis
y_L	Position of the leader along the oy earth frame's axis
z_L	Position of the leader along the oz earth frame's axis

Table 2.2: Aircraft's Available Measures

6 parts: the two on the upper right corner present the aircraft in the oxy and oyz planes, respectively. The corresponding plots on the right and below the latter compare the current position with the target one. And the last one represents the control commands as following: the vertical left command is the thrust τ , the horizontal left command is δ_ζ , the vertical right command is δ_η and the horizontal right command is δ_ξ .

CHAPTER 3

SUPERVISED LEARNING

While fluid dynamics experiments and complex CFD simulations produce more and more data, machine learning algorithms are considered as useful tools for extracting information. These can then be translated into valuable knowledge about the related flows. More than only translate the fluid complexity, it permits to automate tasks for the flow control and optimization. This chapter aims to give a summary of machine learning tools which will be used in this work, namely the supervised learning.

But before going further, let's recall some fundamentals about machine learning. Since this is a very vast domain and a lot of experts has been studied it already, we must give some boundaries to prevent us from getting lost. Machine learning algorithms may classically be divided into three main parts which are the supervised, semi-supervised and unsupervised learning, as presented on Fig. 3.1. While supervised learning algorithms are working ahead of the application on a given dataset, the unsupervised learning ones are constantly learning and searching for improvements. In other words, supervised learning and unsupervised learning algorithms both have the capacity of observing the environment, but the latter are also able to determine a set of actions for an agent which interacts with a specific environment. That is, an agent which uses an unsupervised learning algorithm will directly interact with its surroundings. On the contrary, an agent that uses a supervised learning tool will only be able to observe its environment while it needs another tool which gives it some control commands. Since our objectives are to gain knowledge about an aircraft's environment, and to help that aircraft to describe its surroundings more precisely, the supervised learning tools were chosen for the purpose. In addition to that, supervised learning tools also propose a wide range of different solutions; though a famous library (`tensorflow`) proposes several ways of using neural networks through the `keras` algorithms.

Along this chapter, fundamentals of NNs are given to the reader to understand how they will be further used and what are the related parameters that will improve their performances and why.

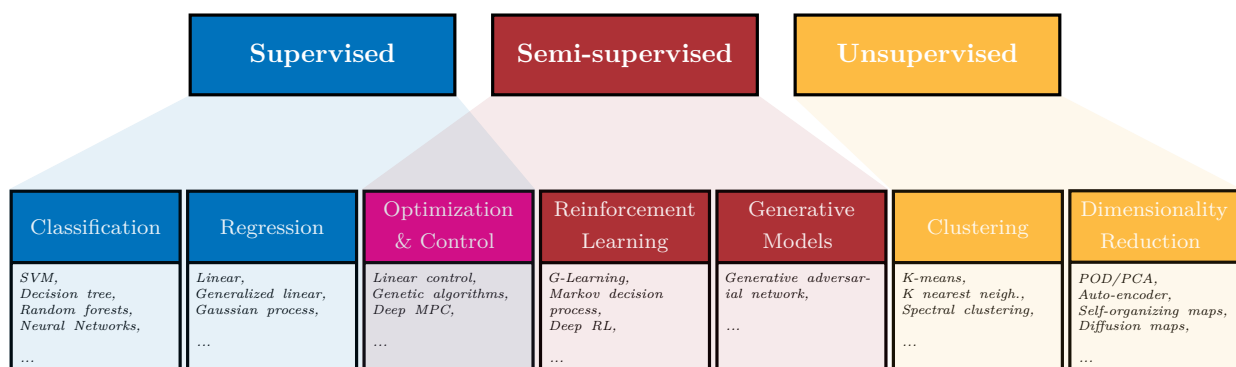


Figure 3.1: Organizational overview of various machine learning methods^[20].

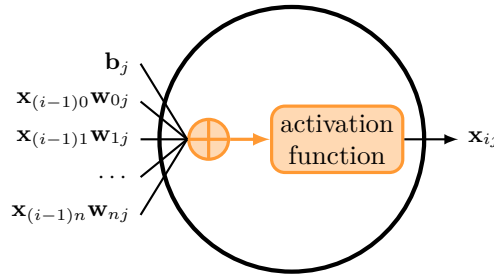


Figure 3.2: A neuron that processes its inputs by passing their sum into the activation function.

3.1 Neural Networks

Neural networks (NNs) were somewhat predesignated for this application. Besides they are the most well known methods in machine learning, the universal approximation theorem states that any function may be approximated by a sufficiently large and deep network^[21]. NNs can be caricatured through the human brain, where each neuron receives an input, processes it through an activation function, and produces an output just as presented on Fig. 3.2. Mathematically speaking, one could write

$$\mathbf{x}_i = f(\mathbf{w} \cdot \mathbf{x}_{i-1} + \mathbf{b}) \quad (3.1)$$

where $\mathbf{x}_i$ are the i^{th} layer's outputs, $\mathbf{w}$ the weights for these, $\mathbf{b}$ the bias and f an activation function. Multiple neurons can then be gathered into a larger structure which will reflect a certain knowledge about the problem and the type of data. These greater structures are presented further, but let's before talk about some other basic elements and concepts such as the activation functions, the loss functions, the optimizers and early stopping.

3.1.1 Activation Functions

Activation functions are determined for layers, meaning that all the nodes belonging to the same layer process their input through the same function. There are several types of activation functions; some of them are presented in Table 3.1. Choosing the right activation function may sometimes become difficult and time-consuming. However several studies have shown that relu was outperforming others; we naturally choose it for the present work^[22].

In the continuous idea of keeping the models' as simple as possible, all layers process their inputs through the same activation function however there is an exception for the output layer: this is indeed aimed to give the prediction which means that we must adapt our choice for the activation function. Therefore, the linear function has a continuous and large range of values so it will be used for regression problems; concerning classification, the softmax function gives the probability for an element to be in one class rather than in any other. Yet softmax function is useful if a single class is expected for each given input. On the contrary, if we are facing a multi-label classification problem, it becomes more interesting to use the sigmoid function.

3.1.2 Loss Function

Furthermore, a reason why machine learning algorithms are gaining in efficiency is because of the nonlinear method, such as back-propagation^[23], used to identify weights which minimize the error between the predictions and the labelled training data. This error is determined regarding the problem we are facing through a *loss function* noted $\mathcal{L}$. A regression problem gives a prediction in a range of continuous values, so referring to the difference between the predicted and the true value is not meaningless

$$\mathcal{L}_i(\mathbf{x}, \mathbf{y}) = \frac{1}{N} \sum_{p=1}^N \|\mathbf{y}_p - f(\mathbf{x}_p)\|^i. \quad (3.2)$$

where $\mathbf{x}$ are the inputs of the network, $\mathbf{y}$ are the expected returned predictions and f the NN itself. Generally, we will use $i = 1$ or $i = 2$ which represent the *mean absolute error* (MAE) and the *mean squared error* (MSE), respectively. Classification problems may use this kind of function but since their predictions are discrete, it is

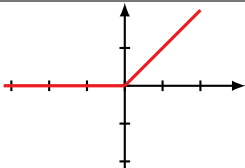
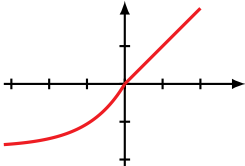
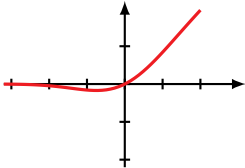
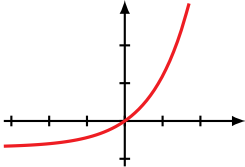
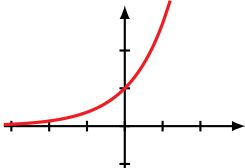
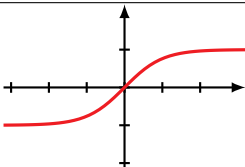
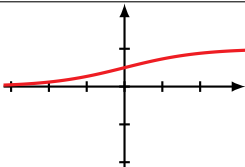
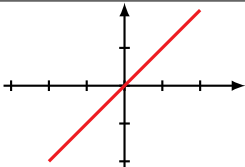
Name	Notation	Expression	Graph
Rectified Linear Unit	relu	$f(x) = \max(0, x)$	
Scaled Exponential Linear Unit	selu	$f(x) = \max(0, x) + \min(0, \alpha(\exp(x) - 1)), \quad \alpha > 0$	
Gaussian Error Linear Unit	gelu	$f(x) = \frac{1}{2}x \left[1 + \tanh \left(\sqrt{\frac{2}{\pi}}(x + 0.044715x^3) \right) \right]$	
Exponential Linear Unit	elu	$f(x) = \alpha(\exp(x) - 1), \quad \alpha > 0$	
Exponential	exp	$f(x) = \exp(x)$	
Hyperbolic tangent	tanh	$f(x) = \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$	
Sigmoid	σ	$f(x) = \frac{1}{1 + \exp(-x)}$	
Linear	linear	$f(x) = x$	
Softmax	softmax	$f(\mathbf{z})_j = \frac{\exp(\mathbf{z}_j)}{\sum_k \exp(\mathbf{z}_k)} \quad \forall j$	

Table 3.1: Activation Functions

not efficient. The loss is then defined according to the entropy of the solution:

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = -\frac{1}{N} \sum_{p=1}^N \mathbf{y}_p \log(f(\mathbf{x}_p)) = -\frac{1}{N} \sum_{s \in \mathcal{S}} \sum_{c \in \mathcal{C}} 1_{s \in c} \log(f(\mathbf{x})_{s \in c}) \quad (3.3)$$

where $\mathcal{S}$ is the set of samples and $\mathcal{C}$ is the set of classes; $s \in c$ represents a sample s that truly belongs to class c . This loss function is called the *categorical cross entropy* (CCE). We effectively observe that less the probability that s belongs to c , greater the loss and worse the accuracy.

3.1.3 Optimizer

Training is the process during which the weights are adjusted to make the predictions as correct as possible. The objective of an *optimizer* is to determine how the parameters of the models are changed, by how much and when. Given the current weights of the model, we compute the related loss function. The latter is telling the optimizer whether it is moving in the right or the wrong direction. Then, it is also of use to determine a learning rate randomly guessed at the beginning ~ 0.01 but decreasing with the number of epochs. The learning rate gives a distance that the optimizer will run for one learning step. While the model is moving towards an optimal position, it could be interesting to diminish the learning rate for gaining in accuracy.

RMSProp We will use for this work two different optimizers; the first one is called the *Root Mean Squared Prop* (RMSProp) and works by keeping an exponentially weighted average of the squares of past gradients. The algorithm then divides the learning rate by this average to speed up convergence. RMSProp is defined by the following set of equations:

$$s_{dW} = \beta s_{dW} + (1 - \beta) \left(\frac{\partial \mathcal{J}}{\partial W} \right)^2 \quad (\text{compute average}) \quad (3.4)$$

$$s_{dW}^c = \frac{s_{dW}}{1 - \beta^t} \quad (\text{correction}) \quad (3.5)$$

$$W = W - \alpha \frac{\frac{\partial \mathcal{J}}{\partial W}}{\sqrt{s_{dW}^c} + \varepsilon} \quad (\text{update}) \quad (3.6)$$

where s is the exponentially weighted average of past squares of gradients, $\frac{\partial \mathcal{J}}{\partial W}$ is the cost gradient with respect to the current layer, W is the weight tensor, β a parameter to be tuned and α the learning rate. The increment ε is a very small value to avoid dividing by zero^[24].

Adam The second optimizer is derived from the RMSProp and is called the *Adaptive Moment Estimation* (Adam)^[25]. First it computes both the exponentially weighted average of past gradients and of the squares of past gradients. Next, it applies a bias correction to both computed values because of their bias towards zero. And it finally updates the weight tensor by using the information from calculated averages. Adam algorithm may be read with the following set of equations:

$$v_{dW} = \beta_1 v_{dW} + (1 - \beta_1) \frac{\partial \mathcal{J}}{\partial W} \quad (\text{compute average}) \quad (3.7)$$

$$s_{dW} = \beta_2 s_{dW} + (1 - \beta_2) \left(\frac{\partial \mathcal{J}}{\partial W} \right)^2 \quad (\text{compute average}) \quad (3.8)$$

$$v_{dW}^c = \frac{v_{dW}}{1 - \beta_1^t} \quad (\text{correction}) \quad (3.9)$$

$$s_{dW}^c = \frac{s_{dW}}{1 - \beta_2^t} \quad (\text{correction}) \quad (3.10)$$

$$W = W - \alpha \frac{v_{dW}^c}{\sqrt{s_{dW}^c} + \varepsilon} \quad (\text{update}) \quad (3.11)$$

This optimizer remains the preferred one. Its gradient descent method showed many its performances over other optimizers such that it always has the first place. Yet it may sometimes give bad results and looking after another optimizer is not forgiven.

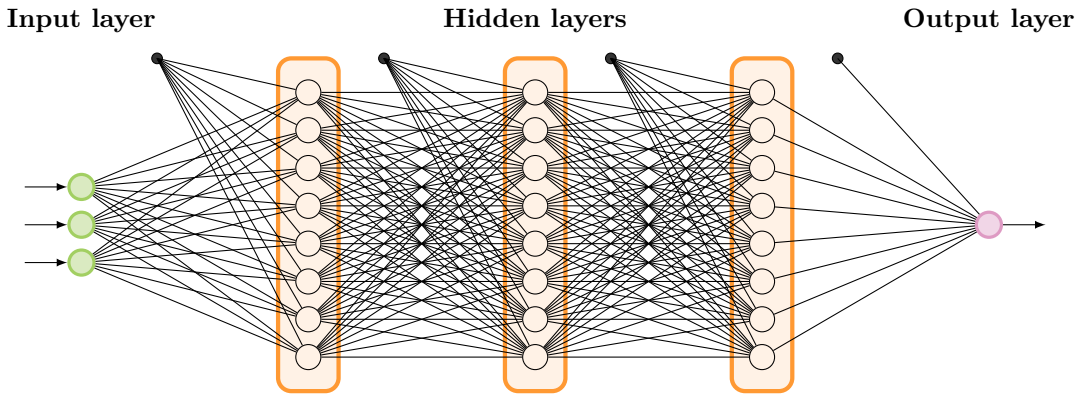


Figure 3.3: Feed-Forward Neural Network

Early Stopping When a model is training, it may get onto a minimum and not learn so much afterwards. It could sometimes even give worse predictions! Therefore we are interested in stopping the learning before the given maximal epoch has been reached. This concept is called *early stopping* and takes the model's loss as a reference point. Then we have the choice about the patience of the model before it activates the early stopping and whether to restore the best weights or to keep the last computed ones.

3.2 Feed-Forward Neural Networks

The most common NNs are undoubtedly the *feed-forward neural networks*, schematically represented on Fig. 3.3; the structure is composed of three types of layers which are the input, the hidden and the output layers. The input layer receives the datas of the network and the output layer produces a prediction. Hidden layers are the main part of the network and are connected to each other from the input layer to the output layer – it only goes in forward direction – and weighted outputs of one layer are the inputs of the next layer.

For the NNs behave just the optimal way, it is recommended to wisely choose some parameters, called *hyperparameters*: we first determines the amount of layers and the amount of nodes on each layer. Although it is possible that each layer has a different amount of nodes, it is more common to see an uniform choice all along the structure. This choice is important since it reflects the complexity of the NN. Indeed, more there are layers and/or nodes, more there are weights to compute. That complexity must be handled because a too simple structure may have a bad accuracy and a too complex one may induce an overfit. *Overfitting* is a doom in machine learning and appears when a model have been overtrained. These models actually seem efficient because their results during training are good but while we are testing them, they show very bad performances. This is the reason why the simpler the NNs' structure, the better it is.

Nevertheless there exist some tricks that prevent overtraining, namely the *dropout* and the *regularizers*. First, a dropout means that the network will neglect a random sample of neurons; these will not be used for the prediction at one turn but could be at the next one. The concept is based on the human brain, with the idea of reducing the codependency during training, which reduces independent capacity of each neuron and overadjust the training data^[26]. Next, the aim of regularizers is about the noise inside the dataset. Overfitting sometimes means that the model is trying too hard to capture the noise in the training dataset. But this is actually not reflecting the reality but rather refers to the spurious small variations of measures. The regularization technique constraints the coefficient estimates towards, or discourage the learning of a more complex/flexible model. The function used for regularization is wheter the L_1 or the L_2 norm which read

$$L_1 = \lambda \sum_i \sum_j |\mathbf{w}_{ij}| \quad \text{and} \quad L_2 = \lambda \sum_i \sum_j \mathbf{w}_{ij}^2 \quad (3.12)$$

for i the layers, j the nodes of respective layers and λ a tuning parameter that decides how much we want to penalize the flexibility of the model. We observe that for a given $\lambda = 0$ the penalty term will have no effect, while it would be maximal for $\lambda \rightarrow \infty$ ^[27].

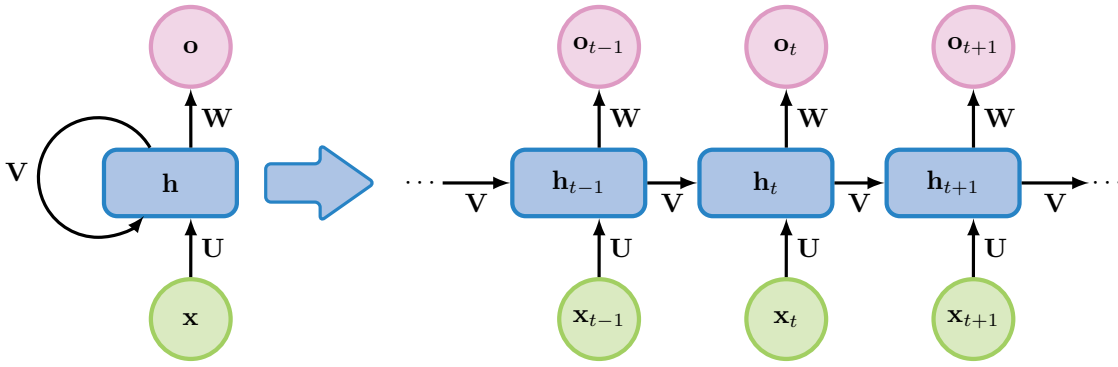


Figure 3.4: One unit recurrent neural network scheme; on the right, the unfold version of the network.

3.3 Recurrent Neural Networks

Apart from feed-forward NNs, it would be interesting to use the fact that the flight path of an aircraft is continuous in time. Indeed this information cannot be easily integrated into a classical NN. This is the reason why using a network which uses the information of past rounds is valuable: these networks are called *Recurrent Neural Networks* (RNNs). In other words, RNNs are neural networks which have recurrent connections. Such networks are composed of interconnected cells and there exists at least one cycle between them. Because of this recurrency, there will be more weights to train since the size of initial inputs is increased by the size of a hidden state: this hidden state is the information that goes from the previous time step to the current one and the output of the cell is also directly dependent of it.

In aim to handle this recurrency, the inputs of the model will gain one dimension with respect to the feed-forward NNs. While we had before one dimension for the batch and a second for the amount of features, we have a third one for the amount of time steps that we want to take into account. Moreover, the cell is different from simple layers because it does not depend on the amount of nodes anymore but rather to the amount of units. The latter is defined as the amount of components we have in the hidden state. In other words, it represents the complexity of the information that goes from one cell to the next one. The behavior of the cell remains similar to the behavior of a simple layer though: at the start, we concatenate the input vector $\mathbf{x}_t$ and the hidden state $\mathbf{h}_t$ and multiply them by the weight's matrix of size $N_{\text{units}} \times (N_{\text{features}} + N_{\text{units}})$. The results of the matrix product then gives a vector which has the same size as the hidden state. This vector is passed into an activation function and it finally gives the output of the cell as well as the next hidden state. One may write this expression accordingly as

$$\begin{aligned} \mathbf{h}_t &= f_h([\mathbf{V} \cdot \mathbf{h}_{t-1}, \mathbf{U} \cdot \mathbf{x}_t] + \mathbf{b}_h) \\ \mathbf{o}_t &= f_o(\mathbf{W} \cdot \mathbf{h}_t + \mathbf{b}_o) \end{aligned} \quad (3.13)$$

where the matrices $\mathbf{V}$ and $\mathbf{U}$ are the weights of inputs vectors $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$, respectively. The size of $\mathbf{V}$ is $N_{\text{units}} \times N_{\text{units}}$ while the one of $\mathbf{U}$ is $N_{\text{units}} \times N_{\text{features}}$. One must pay attention though to the fact that the weights are shared between the cells which allows us to minimize the amount of weights that we have to train! Then, the final resulting output is a non-linear combination of its inputs. A scheme of a RNN and its behavior are shown on Fig. 3.4 in its compact form as well as in an unfold version [20,28].

Their success for natural language processing and speech recognition make them to be efficient for time-dependant problems thanks to a back-propagation through time for obtaining weights. Therefore, they are using same optimisation algorithms as classic feed-forward networks. Nonetheless, some observed that the gradient were vanishing which causes the inability of the network to improve its weights' values. In aim to solve this problem, some improved those simple RNN cells into more complex ones, presented in two next subsections.

3.3.1 Long-Short Term Memory

Although RNNs suffered from vanishing gradient problems during training sessions, they have been largely improved through the development of the *Long-Short Term Memory* (LSTM) algorithms [29]. As depicted on Fig. 3.5a, LSTMs cells share two information's vectors within the recurrency. The first one represents what

matters at short term – it is the hidden state $\mathbf{h}_t$ – and the second one what we mind for the long term – it is the *cell state* $\mathbf{c}_t$. The global behavior of the LSTM may be described into 3 main steps^[28]:

1. We pick up in the cell state all the pertinent information which are coming from the past through the *forget gate*;
2. Then, we choose from the current input which information seems important for the long term through the *input gate*. These are added to the cell state that carry the long term information;
3. Finally, we choose again from the current input which information seems important for the short term through the *output gate*. These are gathered to the cell state which gives the hidden state and the output.

Steps depicted on Fig. 3.5a are gathered in the set of equations hereunder:

$$\mathbf{F}_t = \sigma([\mathbf{V}_F \cdot \mathbf{h}_{t-1}, \mathbf{U}_F \cdot \mathbf{x}_t] + \mathbf{b}_F) \quad (\text{forget gate}) \quad (3.14)$$

$$\mathbf{I}_t = \sigma([\mathbf{V}_I \cdot \mathbf{h}_{t-1}, \mathbf{U}_I \cdot \mathbf{x}_t] + \mathbf{b}_I) \quad (\text{input gate}) \quad (3.15)$$

$$\mathbf{O}_t = \sigma([\mathbf{V}_O \cdot \mathbf{h}_{t-1}, \mathbf{U}_O \cdot \mathbf{x}_t] + \mathbf{b}_O) \quad (\text{output gate}) \quad (3.16)$$

$$\mathbf{c}_t = \mathbf{F}_t \mathbf{c}_{t-1} + \mathbf{I}_t \tanh([\mathbf{V}_c \cdot \mathbf{h}_{t-1}, \mathbf{U}_c \cdot \mathbf{x}_t] + \mathbf{b}_c) \quad (3.17)$$

$$\mathbf{h}_t = \mathbf{O}_t \tanh(\mathbf{c}_t) \quad (3.18)$$

The gates are vectors which size is the same as the hidden state and cell state vectors, it means $N_{\text{units}} \times 1$. Therefore the product of a gate with $\mathbf{c}_t$ or $\mathbf{h}_t$ is done between two corresponding terms such that the result is a vector which size is the same. We observe in this set of equations that a LSTM cell is using two activation functions. In fact, there is an activation function for processing the inputs and a second one that works on the recurrent aspect of the cell. As represented on Fig. 3.5a, the $\tanh$ and σ activation functions are predesignated for this job but could be replaced by other ones. Having the same idea for the dropout and regularizer hyperparameters, one could determine a first one for the kernel and a second one for the recurrent work. Although a LSTM cell is a solution to the vanishing gradient problem, it increases by a lot the amount of weights with respect to the total of units we want in the hidden state. There it may become much more expensive in terms of computational resources.

3.3.2 Gated Recurrent Unit

An alternative to LSTM networks that is cheaper are the gated recurrent unit networks (GRU). Their behavior and performances are comparable to LSTMs networks yet they give the advantage of computing less weights: similarly to the simple RNN cell, only the hidden state is shared between two cells. Moreover, the forget and input gates are gathered into a single gate called the *update gate*. On the other hand, the output gate is replaced by the *reset gate*. A GRU unit cell is shown on Fig. 3.5b and its ruling equations are written below:

$$\mathbf{Z}_t = \sigma([\mathbf{V}_Z \cdot \mathbf{h}_{t-1}, \mathbf{U}_Z \cdot \mathbf{x}_t] + \mathbf{b}_Z) \quad (\text{update gate}) \quad (3.19)$$

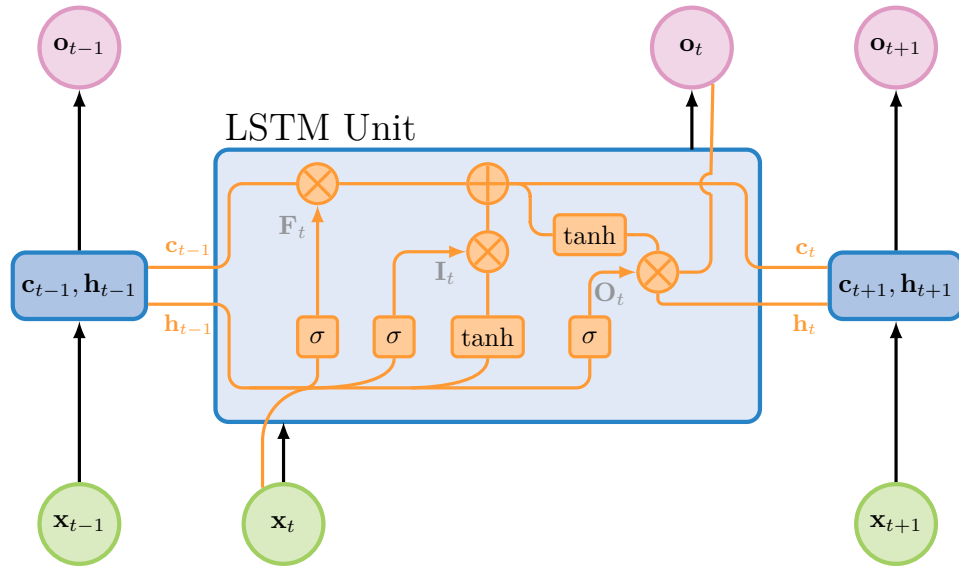
$$\mathbf{R}_t = \sigma([\mathbf{V}_R \cdot \mathbf{h}_{t-1}, \mathbf{U}_R \cdot \mathbf{x}_t] + \mathbf{b}_R) \quad (\text{reset gate}) \quad (3.20)$$

$$\mathbf{h}_t = \mathbf{Z}_t \mathbf{h}_{t-1} + (1 - \mathbf{Z}_t) \tanh([\mathbf{V}_h \cdot (\mathbf{R}_t \mathbf{h}_{t-1}), \mathbf{U}_h \cdot \mathbf{x}_t] + \mathbf{b}_h) \quad (3.21)$$

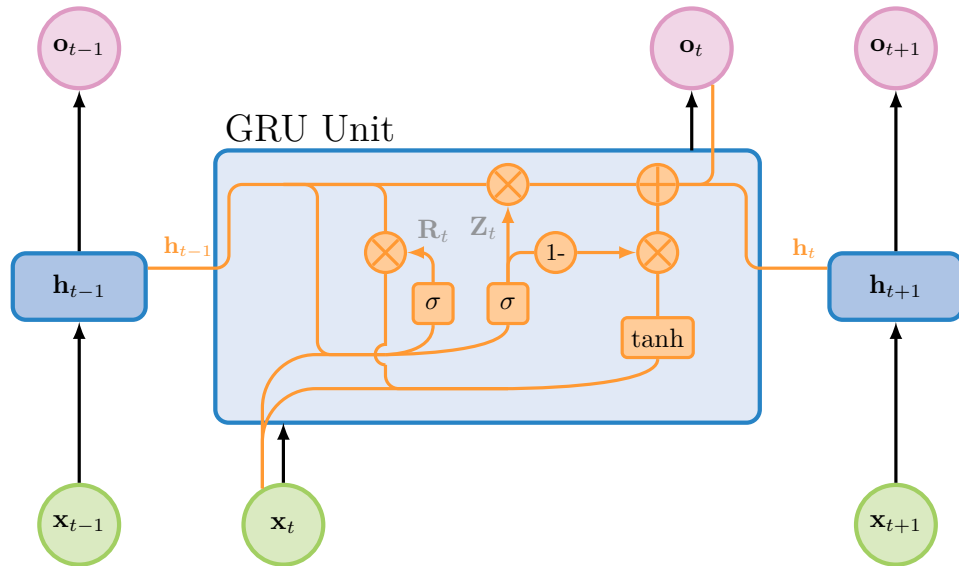
For our purpose, RNNs are thus networks which will use more information than classical networks since they will be aware of what happened previously. Therefore the database will be more important depending on the amount of time steps we want to take into account. Indeed, for each sample trained, we must give the network the set of relevant features of the current time step but also these of the previous time steps we want to use. Then, only the output of the last cell will remain, which is the prediction/approximation we were expecting.

3.4 Building a Model

Now that we have built a proper database, we can go into the next step which consists in building a model. As mentioned previously, we must pay attention to the complexity of the model, so keeping a quite simple model is important. There is hopefully some conveniences which we can use for simplifying the approach.



(a) One unit long-short term memory neural network scheme; operation graph is shown for time step t and weights are not shown.



(b) One unit gated recurrent unit neural network scheme; operation graph is shown for time step t and weights are not shown.

Figure 3.5: One unit of improved recurrent neural networks schemes.

3.4.1 Normalization

First of all, normalizing data is a very common thing in machine learning. All features are not laying in the same range of values; it means that if some of them have a much larger range of values compare to others, they get more chances to have an impact during the training. This is the reason why we want to equalize those ranges in the form

$$X' = \frac{X - \mu}{\sigma}. \quad (3.22)$$

where μ represents the mean value and σ the standard deviation. The features have now a zero-mean and approximately 70% of their values laying in the range $[-1, 1]$. We expect with such a database that each features has an equal impact at the start of the model's training.

3.4.2 Hyperparameters

Next, we saw that models have some hyperparameters which determine their behavior. Depending on the global structure of the network, the type of hyperparameters will change. As discussed earlier, we decide for this work to work with feed-forward NNs and with RNNs. The related hyperparameters are described in the next few paragraphs.

Amount of Layers/Cell Since feed-forward NNs behave differently from RNNs, we will not build them the same way. Concerning feed-forward NNs, it is not rare to determine several layers, starting from 2 to a maximum of 5. More the layers, more complex the model, and we must pay attention to the risk of overfitting. This is the reason why for a small amount of layers, we will rather set a high number of nodes, and conversely when the amount of layers increases. On the contrary, a unique RNN cell is generally sufficient for an efficient work. Furthermore, it is of use for both structures to add an output layer in aim to process the final prediction correctly as discussed in Section 3.1. Concerning the recurrency of RNNs, one care to keep it as low as possible since increasing them will need more space in the computer's memory and it is more expensive in terms of computational ressources. One still expects that a maximum of 5 time steps is sufficient for obtaining good performances.

Amount of Nodes/Units We decided for a simplified understanding to use an uniform amount of nodes in each layer of a feed-forward NN. This amount is commonly determined by the values 2^n , for $n \in \{3, 4, 5, 6, 7, 8\}$. We consider that other values are meaningless for further reasons: if there are less nodes, the network would become too simple or whether we have to increase the amount of layers which decreases the understanding. If the amount of nodes is higher, there are too much weights and the risk of overfitting becomes too high. Concerning other values laying between them, it is not so interesting to study a model's behavior for any amount of nodes. Being limited to 2^n values gives a wild range while keeping the amount of studied models quite small. Besides, the amount of units in RNNs will follow the same scheme for the same reasons.

Activation function For the same reason of simplifying the understanding, the decision was made to use a common activation function for all the hidden layers. From previous discussion about them, we enlightened the performance of relu function which will be used in all hidden layers. Concerning the RNNs, we will rather use the relu function on the recurrency aspect when it is possible, it means for LSTM and GRU cells. Since simple RNNs process their inputs through a single function, we choose it to be relu as well. Furthermore, we also discussed about the activation function for the output layer in Section 3.1 as the softmax function is a the right choice for classification problem and the linear function for regression problems.

Dropout The dropout parameter is the percentage of weights that will be ignored for the turn. Its value lays then in the range $[0, 1]$, 0 meaning that all weights will be considered and 1 for none of them. We can expect that the performances will decrease if we have a large value, but the risk of overfitting is present for a low value. Keeping a dropout between 20 and 50% seems then reasonable. For classification problems, we will then use a 50% dropout after each hidden layer or a recurrent dropout for RNNs. Concerning the regression problems, the idea is the same with a 20% dropout (resp. recurrent dropout) for feed-forward NNs (resp. RNNs).

Regularizer For a higher penalization of high valued weights, we prefer to use the L_2 norm regularizer. Concerning the tuned parameter λ , it is common to keep it low, below 1. We decide to initialize it to $\lambda = 0.01$ regarding standard initialization.

3.5 Training and Validation

Even if the way a model has been built is the most important in aim to have good performances, the `tensorflow` library offers plenty way of improving the training. In fact, not choosing a good way may take much more energy if not decrease the performance of a model.

3.5.1 Loss Function

Just as described in Section 3.1, we will use different types of loss function for classification than for regression problems. Classification problems will gain efficiency with the categorical cross entropy function which takes the probability of the model to predict the right class. More the probability is weak, more the loss is important. Furthermore we will use a second metric for deepening the understanding of our future classification models which is the *accuracy*. The accuracy is the probability that the model predicts the right class. Indeed we could sort the prediction of a 2-classes problem as presented in the following table:

	True Positive	False Positive
Predicted Class True Class	Positive Positive	Positive Negative
	False Negative	True Negative
Predicted Class True Class	Negative Positive	Negative Negative

The accuracy is then given by

$$\text{accuracy} = \frac{\text{Amount of correct predictions}}{\text{Amount of total predictions}} = \frac{TP + TN}{TP + TN + FP + FN}. \quad (3.23)$$

Concerning the regression problems, we already announced that we will work with both MSE and MAE metrics. However one of them is more appropriate for being used as the loss function but it will depends on the type of problem we are facing. If the solution of the regression problem is expected to be quite large, the error between the predictions and the true values could be large too; using the MSE loss function with such a problem will considerably increase the error while the MAE values would be more accurate. On the contrary, if the range of expected values are small, it becomes more interesting to use the MSE because small errors would become even smaller!

3.5.2 Optimizer

The Adam optimizer is surely the most used of all since it has rapidly shown good performances^[25]. However, even if β_1 and β_2 parameters are tunable, it is common to keep them higher than 0.9 because of their exponentially decaying. Indeed, they are multiplied by themselves at each step such that a small value could make the learning much slower! For that reason and for simplified use of the Adam optimizer, the decision was made to keep their default values, meaning $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

Nevertheless training showed that the Adam optimizer was not so good for classification problems, and that the RMSProp algorithm was the solution for. With the same idea as for the Adam algorithm, the default value of $\beta = 0.9$ is kept which avoid us to train much more models in aim to find the optimal learning.

3.5.3 Validation

The last element of training process is called the validation. What the matter how good the model is performing during the training if it cannot make a good prediction on new values ? This is the reason why the database is

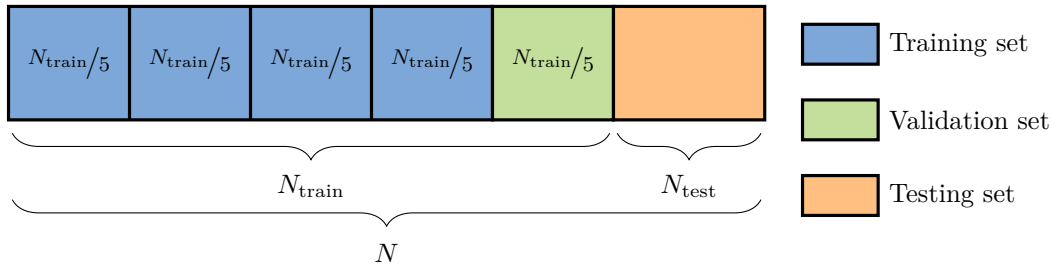


Figure 3.6: Database Subsets

generally splitted into several parts as presented on Fig. 3.6. Before any training, we split the database into a train and a test set; the usefulness of the latter will be discussed in the next section. The data from the train set are used for training the model. But while we are adjusting the weight tensor, one could try to know the current efficiency of the model on data he did not train with. This is the validation set. In fact, if we split the train set again into several subset, the model could use each one of these as validation set and see its performances. It is common to split a set into 5 equal parts, as it gives a reliable range of values. Too small sets could not be reliable because of the small range of values inside, but splitting the training set into two parts for example makes the training less efficient and much slower.

3.6 Testing

Finally, the model has been trained and showed good performances. We compared several of those with different parameters and we select the best one. We want now to verify how efficient it is. For that purpose, we would want completely new data. The testing part will happen in three parts: first, a last validation of the model when it is built, second the construction of a new database for testing, and a third one for analyzing the model's error. The very first refers back to Fig. 3.6: remember that we splitted the database into two parts: a training and a testing set. The data from the testing set has not been used during the training so verifying the model a last time before using it is not meaningless. It permits us to verify at least if no overfitting was made and to select another model or make some adjustments on the selected one. This is even more supported by the fact that validation occurs at the same time as training and is a result used for model's selection. Therefore the test is performed on the chosen model only in aim to verify its performances a last time before using it.

Next, we build a new database as we did for the training (explained later in Section 4.1) but with a larger amount of data. In that respect, we will have a big sample and more accurate results. Tests on the error may need lots of data so being able to enlarge the set is a good point. And at last, we will use knowledge about probabilities and statistics for a deep analysis of the model's error.

Now that we have a better idea of what is the matter of this work, we can go deeper with presenting the models. These must help an aircraft to know better its environment, and to localize a wake with precision. Since a wake may be assumed to be low-decaying and infinitely long in the ox direction, we do not want him to go back and forth all the time. Therefore the flight path is limited to the oyz plane but the aircraft is not constraint along the ox axis, because we care much more about its stability.

Then, how can we characterize a wake ? And what information could be useful for a follower aircraft ? Remind that a wake can be very dangerous and cause dramatic damages. For that reason, it seems essential to know where are located surrounding wakes with respect to the follower aircraft. Yet the localization is important, it may also be interesting to know the intensity of vortices and the distance between them. Approximating the latter corresponds to characterizing the leader aircraft, because it is the one which creates the wake. Since we are assuming that both the leader and the follower are the same type of aircraft, it seems unnecessary to go further in that direction. Therefore, we focus our mind on the wake's localization.

This chapter first aims to present the way that we built the database on which the models will be trained. Next, we will go through the different models that have been studied, by determining for each one of them both specifications and expectations.

4.1 Database

Let's then first build our database: since it is more common used to fly in the wake of a single leader, we define at first an environment with a single wake at the origin of the oyz reference plane. The idea is to make a plane flying around and record as much data as possible. The more these data are different from each other, the better it is because algorithms may then be able to target the most reliable characteristics. In that respect to have a wild range of features' values, we decided to set up a randomized target policy. On the counterpart, it seems not interesting to have an aircraft flying very far from the wake since its intensity decreases exponentially with the distance: we prevent it by limiting the flight path in a box of given size. The database has then been built with respect to the following steps:

1. The environment is defined with a single external plane, placed at $(y_L, z_L) = (0, 0)$ and having the same characteristics (Γ_0, b_0) as the studied plane.
2. The simulation box is defined such as the flight path is limited by

$$y_{\min} = -3b \leq y \leq 3b = y_{\max} \quad \text{and} \quad z_{\min} = -\frac{3}{2}b \leq z \leq \frac{3}{2}b = z_{\max} \quad (4.1)$$

3. The simulated aircraft is initialized with a positional target policy at a random position in the ring circle of radius

$$\frac{b_0}{2} \leq r_i \leq \min(y_{\max}, z_{\max}) \quad (4.2)$$

and angle θ_i . Its position is then given by

$$x_0 = 0, \quad y_0 = r_i \sin \theta_i \quad \text{and} \quad z_0 = r_i \cos \theta_i. \quad (4.3)$$

The velocity is zero and the target is its own position. It means that it will not move from that initial position if it is not told to.

4. Simulation is started, and after a time offset $t = 10$ [s], the target is modified following:

- A new target is chosen inside the ellipse which equation is

$$\left(\frac{y_{\max} - y_{\min}}{z_{\max} - z_{\min}} \right)^2 (y_T - y)^2 + (z_T - z)^2 = b^2. \quad (4.4)$$

In that respect, the aircraft is not able to “jump” above the wake because the target is on the other side.

- This new target is kept if it lies inside the simulation box and if it is far enough from wakes. Otherwise an other target is chosen.
 - After 100 iterations, if no suitable target has been found, the simulation begins back at its initial state.
 - A new time offset is randomly chosen in the range [5, 15] seconds.
 - A schematic representation of possible target positions are represented on Fig. 4.1.
5. At each time interval $\Delta t = \frac{1}{2}$ [s], states, measures, control and predictions are stored in a vector.
6. When simulation time reaches $t = 1000$ [s] ≈ 16.67 [min], then the simulation is stopped and relevant data are stored.
7. For the sake of stability, we do not want that the follower aircraft comes too close to the leader’s wake. Therefore, we verify at each time step the distance between them both. If we observe that

$$(\Delta y_L)^2 + (\Delta z_L)^2 < \left(\frac{b_0}{2} \right)^2 \quad (4.5)$$

then the simulation begins back at its initial state. Even if the target policy has taken this into account, the PID controller might decide to go through as well. This is that special case which we are interested in here.

! Pay attention though: even if the simulation begins back, all recorded data are kept. The new simulation is then shorter since we want a total simulation time of 1000 [s].

This database is key for an efficient training. As for any application, trying to do something with bad or useless information could bring more bad effects than any positive expected results. In that respect, a clever analysis of the problem is necessary in aim to obtain efficient models. We saw in Chapter 2 that the aircraft dynamics is depending on its environment. More it is close to a leader wake, more it will be subjected to induced velocities and feels the corresponding accelerations. It would then be interesting to evaluate the speed differences induced by the environment on the aircraft dynamics. This is the reason why we decided to compare two aircraft which have same positions, velocities and control commands and then to substract their aerodynamical efforts. The idea is presented on Fig. 4.2 where we observe that the isolated aircraft has a different acceleration vector than the follower. There are actually 8 measures that might be different from one aircraft to the other: $\ddot{x}$, $\ddot{y}$, $\ddot{z}$, $\ddot{\phi}$, $\ddot{\theta}$, $\ddot{\psi}$, α and β . To this end, we will use the difference of these measures which read

$$\Delta m = m - m_{\text{isol}} \quad (4.6)$$

where m represents a random measure of these presented above. Since it seems obvious for accelerations to be different for the both aircrafts, it might not be for the angle of attack and the angle of slip. Nevertheless, they are directly related to the wind induced speeds which are dependent from the environment. Having a wake in the surroundings of an aircraft will then have an impact on these two angles too! Some other measures could be

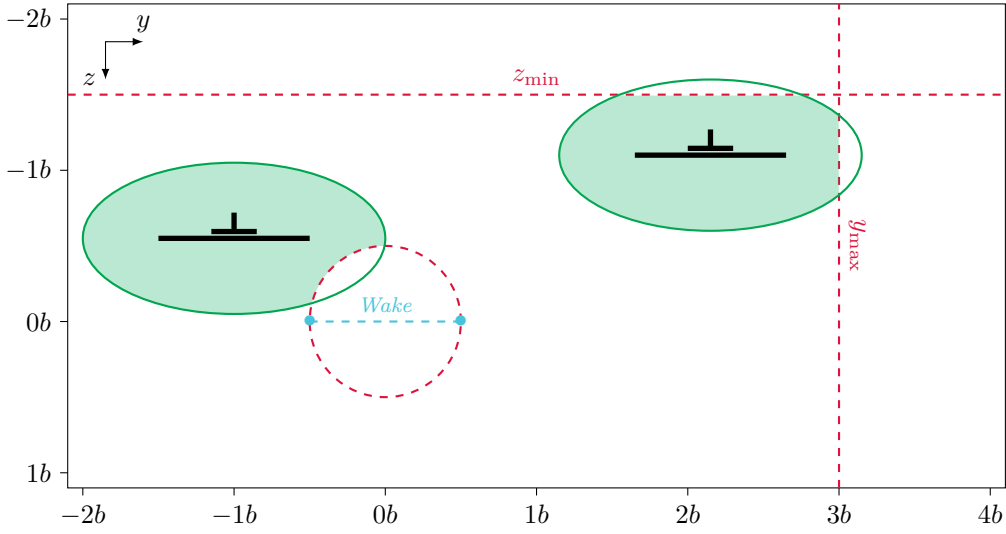


Figure 4.1: Target policy for building a suitable database in aim to record relevant data for training and testing the learning models. The green ellipses represent the range of random selected targets. Sometimes the ellipse crosses undesired areas: therefore we pick another random target if the chosen one is not inside the green filled area. The case is presented on the figure for two different aircraft's positions.

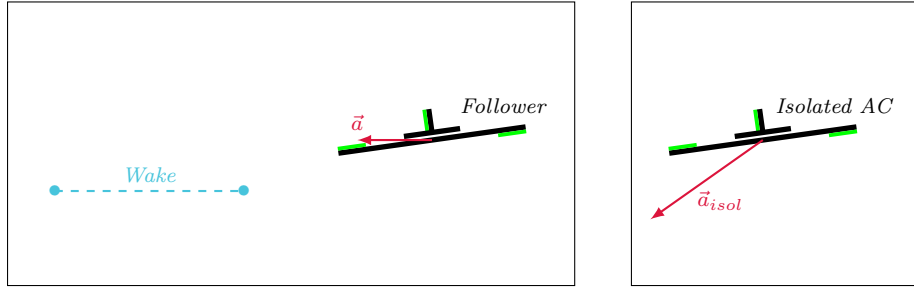


Figure 4.2: Feedback of the environment on an aircraft. At left, the aircraft is subjected to the induced speed of a leader's wake while at right, the wake is not taken into account. Although these aircrafts have the same positions, velocities and control commands, their aerodynamical efforts are different.

interesting for more advanced models but then recording the true aircraft's measures only would be sufficient.

A last step in building the database consists in computing the objective of the corresponding model. In supervised learning, training a model consists in attributing some weights when starting from chosen features, we can determine as best as possible the wanted result. During training, we must then have for a set of features the corresponding results that the model is expected to obtain. Only then it will be able to compute efficient weights and give accurate results. That last step is related to what we are expecting from the model and is discussed in the next section.

4.2 Specifications

The localization of a wake can be splitted into two parts: an orientation with respect to the follower aircraft, and a distance between them. Both of them are defined as

$$\chi = \arctan \left(\frac{\Delta y_L}{\Delta z_L} \right), \quad (4.7)$$

$$d = \sqrt{(\Delta y_L)^2 + (\Delta z_L)^2}. \quad (4.8)$$

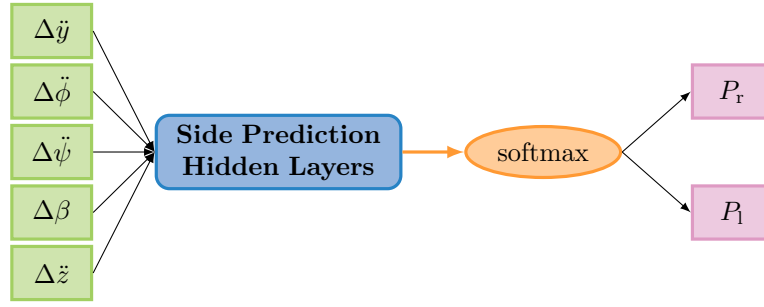


Figure 4.4: Inputs and Outputs of the Side Prediction Model.

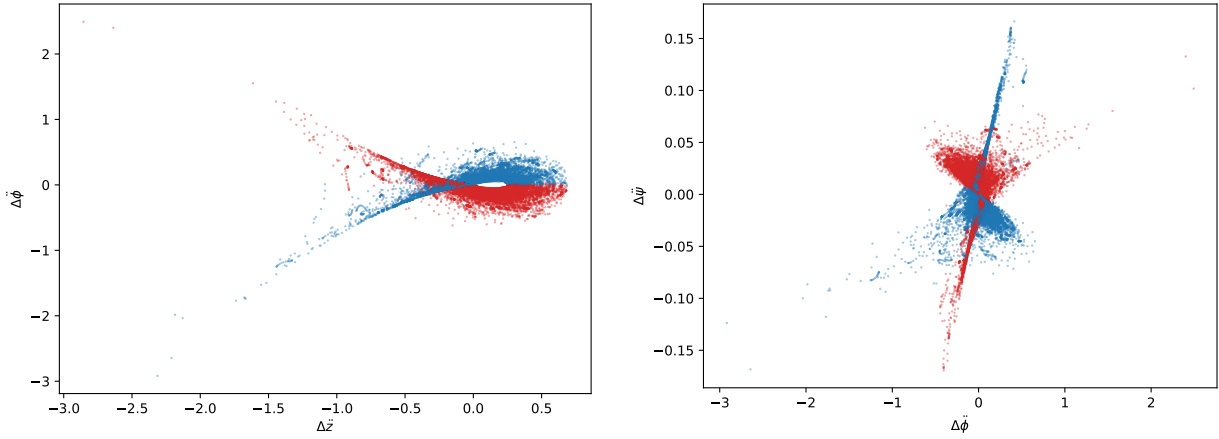


Figure 4.5: Database for predicting on which side a follower aircraft could potentially find the wake of a leader – in red, the wake is starboard, in blue it is portside.

probabilities of finding a wake on starboard and portside, this is a classification problem. In addition to that, we expect a data to correspond to one case because a wake could not be located on both sides, simultaneously. Therefore the activation function of the output layer is chosen to be the softmax function. We observed earlier that for using the symmetrical characteristic of the wake, we must first analyse its non-symmetric behavior. Non-symmetric measures would therefore be useful for predicting on which side we have higher probability to find a wake. Furthermore, we observed that (oxz) was actually the plane of symmetry; since the same plane has been used for the longitudinal aircraft dynamics (confr. Section 2.4), we know for sure that measures related to that dynamics are symmetric with respect to wake. On the contrary, measures related to the lateral dynamics are non-symmetric, which means that values starboard and portside are opposed in signs. A reminder of related measures is proposed hereunder:

$$\text{Symmetric measures: } x, z, \theta, \alpha, \dot{x}, \dot{z}, \dot{\theta}, \text{ etc.} \quad (4.9)$$

$$\text{Non-symmetric measures: } y, \phi, \psi, \beta, \dot{y}, \dot{\phi}, \dot{\psi}, \text{ etc.} \quad (4.10)$$

In the end, we mentioned in Section 4.1 that only the difference of aerodynamic efforts were useful in the detection of a leader's wake. Therefore we are interesting for this model to use the following measures: $\Delta\ddot{y}$, $\Delta\ddot{\phi}$, $\Delta\ddot{\psi}$ and $\Delta\beta$, where the Δ represents the difference between the real aircraft and a second one for which the environment has not been taken into account. Unfortunately, it has been observed that models presented bad performances and it seems not meaningless to add the acceleration differences along the oz axis. Even if vortices are upwashing on both sides of the wake, they are hardly downwashing at the center; this difference that the non-symmetric measures cannot notice is valuable for determining the side when the aircraft is over or below the wake. Yet, it has not really practical use in reality because aircraft's are not supposed to fly there, it may prevent from danger if detected! A summary of what has been said about the side prediction model is presented on Fig. 4.4.

Measure	$P_r > P_l$	$P_r < P_l$
$\Delta \ddot{x}$	$\times(1)$	$\times(1)$
$\Delta \ddot{y}$	$\times(1)$	$\times(-1)$
$\Delta \ddot{z}$	$\times(1)$	$\times(1)$
$\Delta \ddot{\phi}$	$\times(1)$	$\times(-1)$
$\Delta \ddot{\theta}$	$\times(1)$	$\times(1)$
$\Delta \ddot{\psi}$	$\times(1)$	$\times(-1)$
$\Delta \alpha$	$\times(1)$	$\times(1)$
$\Delta \beta$	$\times(1)$	$\times(-1)$

Table 4.1: Preprocessing Factor for the Orientation Approximation Model.

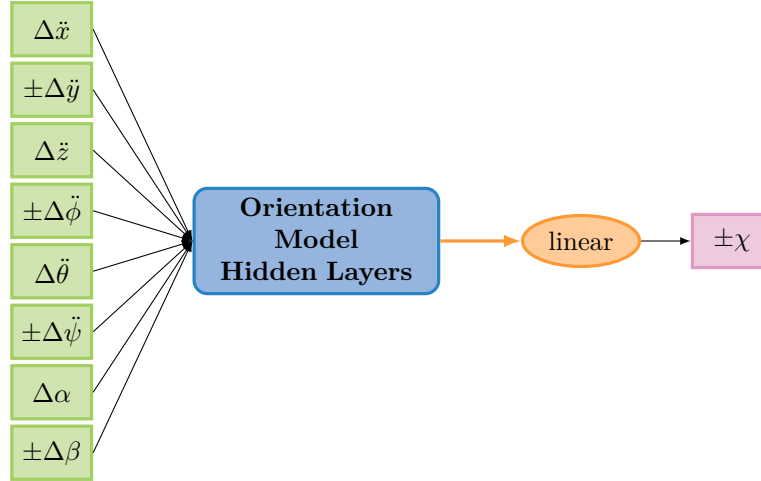


Figure 4.6: Inputs and Outputs of the Orientation's Approximation Model.

Next model in the list is the approximation of the wake's orientation. Since we are now able to say on which side there is a wake, one may gather measures as if we suppose that the aircraft only flies portside to the wake. Whereas predicting the side was a classification problem, evaluating the orientation of the wake is a regression one. Given that, we will use the linear activation function for processing the output layer. All difference of aerodynamic efforts, as well as differences of the angle of attack and the angle of slip are expected to be significant for the purpose. Yet we changed the frame with respect to the symmetry, we must preprocess these inputs according to Table 4.1. Since the prediction of the model is also symmetrical with respect to the wake, it needs to be processed to the side model's prediction too! A first sketch of this is presented on Fig. 4.6.

The third and last model consists in predicting if the follower aircraft is getting closer or not to the wake. Because of that, it is a single-label classification problem which means that the softmax function will be used for processing the outputs. As for the side prediction model, the output will represent the probability of the follower aircraft of getting closer, or of going further. As far as the matter of problem is about the aircraft's displacements, differences induced by the environment seems not significant anymore. Therefore, we will use velocities of the lateral dynamics as well as the orientation of the wake: v , w , $\dot{\phi}$ and χ . Nevertheless, it seems irrelevant to build such a model since computing the approach along a line is just a matter of trigonometrics: in fact, if the difference between the direction of the wake and the direction of the displacement lays in the range $]-\pi/2, \pi/2[$ then we are getting closer:

$$\left| \arctan\left(\frac{v}{w}\right) - \chi \right| < \frac{\pi}{2}. \quad (4.11)$$

The influence of the roll rate is not perceptible yet it is part of the lateral-direction dynamic. However, for the sake of simplicity, we will not consider it as being a valuable information for this model. One may tackle the problem as being a matter of vortex intensity as well: more the follower comes close to the wake, more aerodynamic effects are important. There we may use the difference in efforts that the follower is subjected to as valuable information. In other words, the efforts induced by the environment have changed between the discrete times t and $(t+1)$. This being said, we will use the acceleration along the gravity axis and the acceleration in

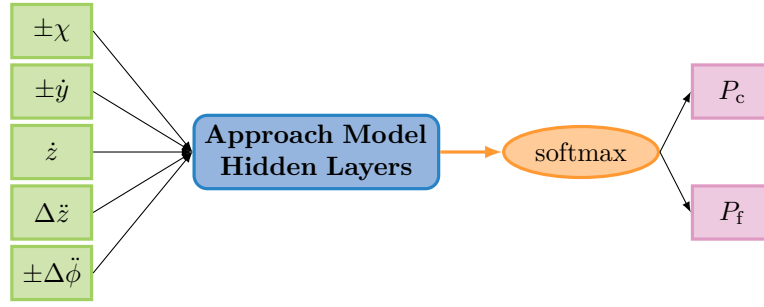


Figure 4.7: Inputs and Outputs of the Approach Predicting Model.

roll to use both dynamics while keeping the amount of features relatively small. An overview of the model is presented on Fig. 4.7.

Finally, one may still wonder why the use of a learning model is needed for this task. Indeed, the both way we faced the problem are predictable and easily hand-done. One could then compute the approach based on these both information. Yet, it is not realistic because of two reasons: at first, there is another model that approximates the direction of the wake. The regression implies an error which could not be known everytime. On the other hand, we recall that the intensity's variations are highly non-linear. The use of NNs then gives the advantage of extracting the right information and balancing the weights that both methods will have on the prediction.

4.3 Expectations

The aim of this section is to sketch the first expectations of the described models. As they have been built, one may represent them into a larger plan, as presented on Fig. 4.8. We observe that the prediction of the side model is used by both the orientation approximator and the approach predictor. In addition to that, we need a good approximation of the orientation in aim to predict the approach correctly. Therefore, the expected accuracy for the side prediction must be very performant because the orientation prediction directly depends on it. The model will then be considered as being efficient if its accuracy is above 90%.

Concerning the approximation of the orientation, the model needs performance in aim to predict the approach afterwards. Yet we decided to use two different techniques for that matter. Nevertheless this approximation is a regression problem so we seek to minimize the error by keeping the angle between the true value and the prediction as small as possible. Since it is a regression problem, we are rationally expecting that the error will follow a zero-mean probability function, in the range $]-\pi, \pi]$. What has to be determine is the type of probability function and its standard deviation σ_χ which will describe how the error is distributed.

In addition to that, one discretizes the range inside which the error lays in aim to ease its description: if the absolute value of the error is smaller than $\pi/4$ radians, an accurate result for the approach (confr. Section 4.2) is still possible. Conversely, if the error gets higher, it means that the models loses much of the information. Finally, if the error is greater than $3\pi/4$ yet smaller than π radians, it results a completely loss and the model cannot be trusted at all. For the latter it is clearly preferred to keep the error always below $\pi/4$ such that we need σ_χ to be smaller. Because of the definition of the standard deviation, we are expecting that the $\text{MSE} = 10 \text{ [deg]} = \pi/18 \text{ [rad]} \approx 0.17 \text{ [rad]}$.

Finally, one may be less regarding for the approach prediction yet it represents valuable information. Unless previous models are very performant, we could not expect very accurate results from this one. Indeed, because of the range of values for its prediction, a regression model might give wildly spread errors. Therefore, the accuracy of the approach's prediction will be lower than what we will observe during training. It is then rationally expected that training performances will be around 90% and that the testing performances after considering the side prediction error would be around $(90\%)^2 = 81\%$.

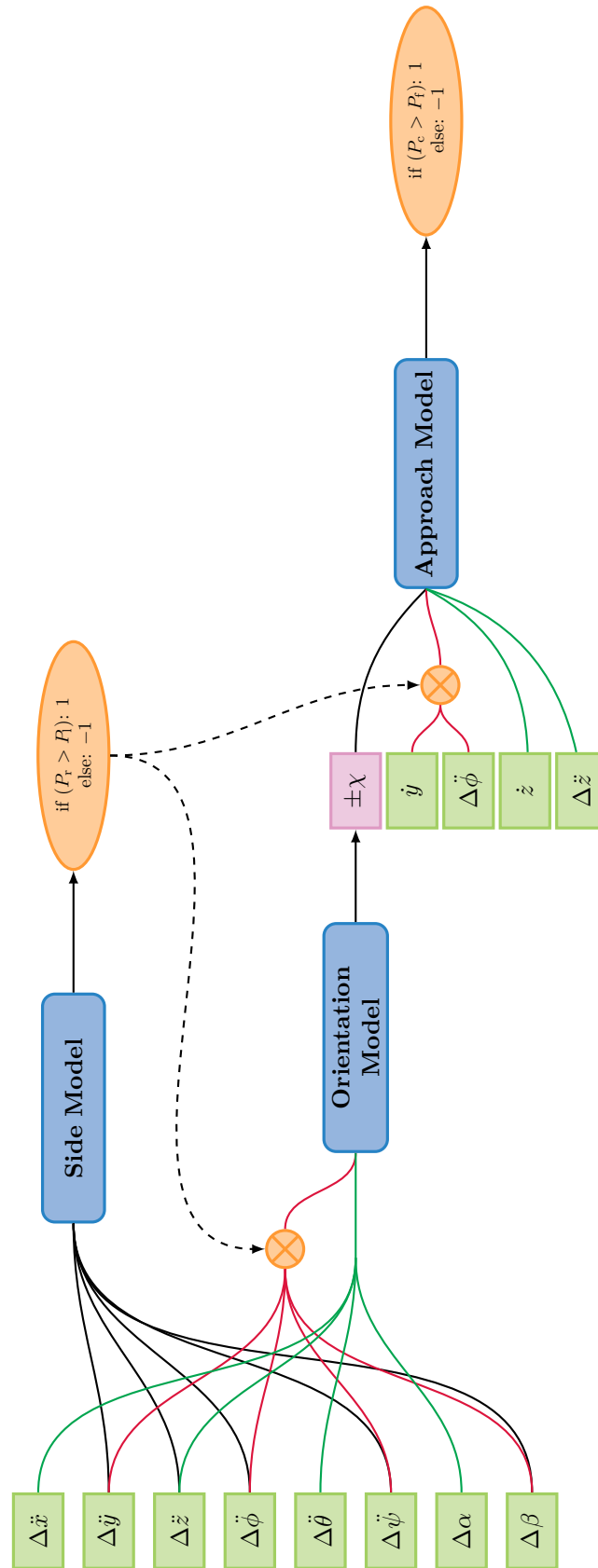


Figure 4.8: Global view of all models, including interconnections and preprocessing phases.

Previous chapters gave the reader some insight to the aircraft dynamics as well as to machine learning algorithms. Our purpose is to build learning models which are able to accurately characterize an aircraft's environment. This chapter aims to compare the training performances of different models, choose the one which showed the best results, and observe if it performs the same in simulation. Then, we will characterize the resulting errors and discuss their performances.

First of all, we are interested in the classic feed-forward NNs: since they are simple structures, they represent an easy way of evaluating the complexity of a problem. They will give a first idea about our expectations drawn in Section 4.3. Next, we talked in Section 3.3 about RNNs and why they could be useful for temporal series problems. RNNs' results are thus compared to those obtained with feed-forward NNs. In addition to that, one want to compare models with respect to several criterias, therefore we will use the different measures used during the training session: CCE and accuracy for classification problems, and MSE and MAE for regression problems.

This chapter is splitted into three sections which means that there is one section for each faced problem: side prediction, approximation of the orientation and approach prediction. These sections are themselves splitted into two subsections for training and testing performances. Finally, we will close the chapter with a discussion about models' performances.

5.1 Side Prediction

5.1.1 Training

We recall that predicting the side of a leader aircraft's wake with respect to the follower is a classification problem which prediction is whether right or left. Therefore, we use the CCE loss function, yet the accuracy represents an easier way to visualize performances. We first trained feed-forward NNs on the presented dataset (Section 4.2) which results in plots on Figs. 5.1. We observe that accuracy expectations are achieved starting from 32 nodes per hidden layer. Now, we can even expect more from that classification model in front of performances for 2 and 3 hidden layers. Indeed, we observe that 1 hidden layer is not enough and that the model could still increase its performances. We furthermore observe that it tends to blow up if the amount of hidden layer increases. In addition to that, one notes that the CCE loss for 2 hidden layers is smaller than for 3. Besides that, performances still increase with the amount of nodes and validation performances are slightly better than for training. That last observation could be explained because the validation set is smaller than the training set such that there could be a difference in range of represented data. One rationally concludes that models with 2 hidden layers which each has 128 nodes are satisfactory.

One may then observe performances of simple RNNs trained on the same problem on Figs. 5.2. On the figure, the label "recurrence step(s)" refers to the amount of time steps that the network is feed with. It means that the blue line do not use recurrence at all! We directly observe that RNNs largely improve performances which increased

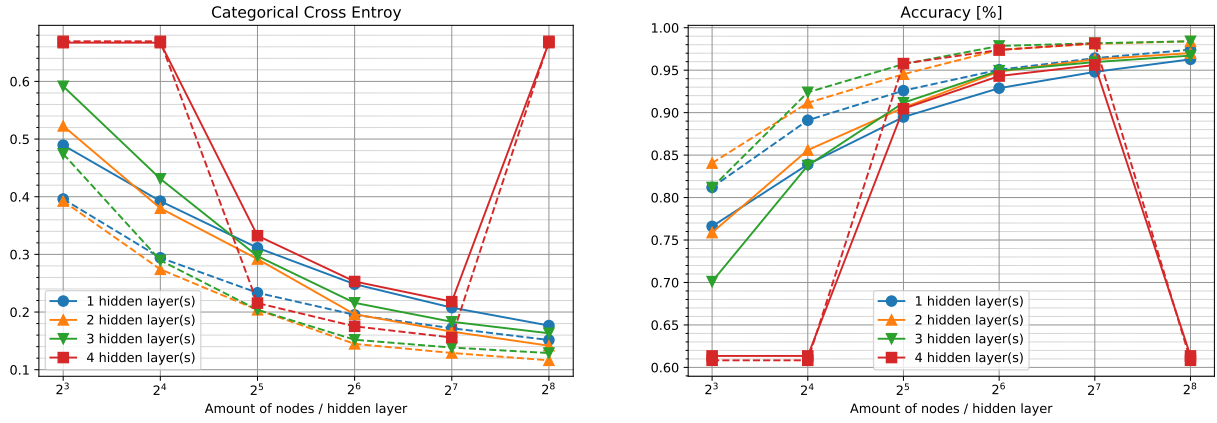


Figure 5.1: Training performances of feed-forward NNs which aim to predict the wake side with respect to a follower aircraft. These are represented for different amount of hidden layers at certain amount of nodes per hidden layer: plain lines (—) represent training performances while dashed lines (- - -) represent validation performances.

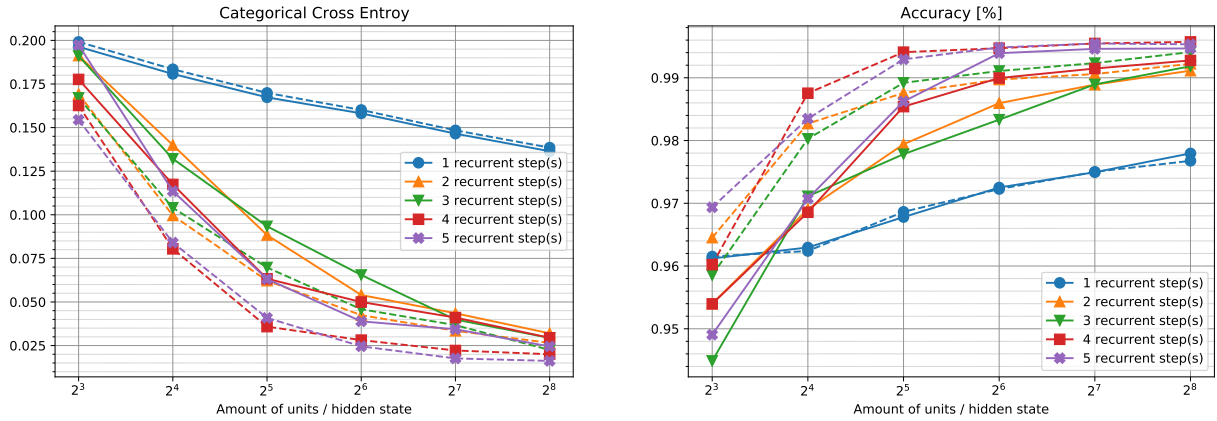


Figure 5.2: Training performances of simple RNNs which aim to predict the wake side with respect to a follower aircraft. Performances are represented for different amount of nodes in recurrent cells in function of the amount of previous time steps which are taken into account: plain lines (—) represent training performances while dashed lines (- - -) represent validation performances.

up to 99% for higher amount of units in hidden state. And regarding to the blue line, it is now clear that the temporal information has an impact on the models' performances. In addition to that, one could also remark that performances are increasing with the amount of previous time steps that we are taking into account. Yet it shows better results and could even improve them, it needs much more computational resources: increasing both the recurrency and the amount of units may actually become very expensive! In addition to that, we also remind that the probability of overtraining is increasing with the complexity of the model. With comparison to feed-forward NNs, we still observe that the models' performances on the validation set are better and again we rationally expect that it does not cover all the features' range of values.

We learned before that simple RNNs were not so efficient because of the vanishing gradient problem which is why we also talked about LSTM and GRU alternatives. Hence we trained these for which you can find performances on Figs. 5.3 and Figs. 5.4, respectively. Again, we observe that taking the time as information consistently increases the performances even though we observe much more accurate results than with simple RNNs. While LSTM models seem quite constant with increasing amount of units, GRU models are largely improving. It also seems that increasing the amount of previous steps which are taken into account will improve performances and give more accurate results.

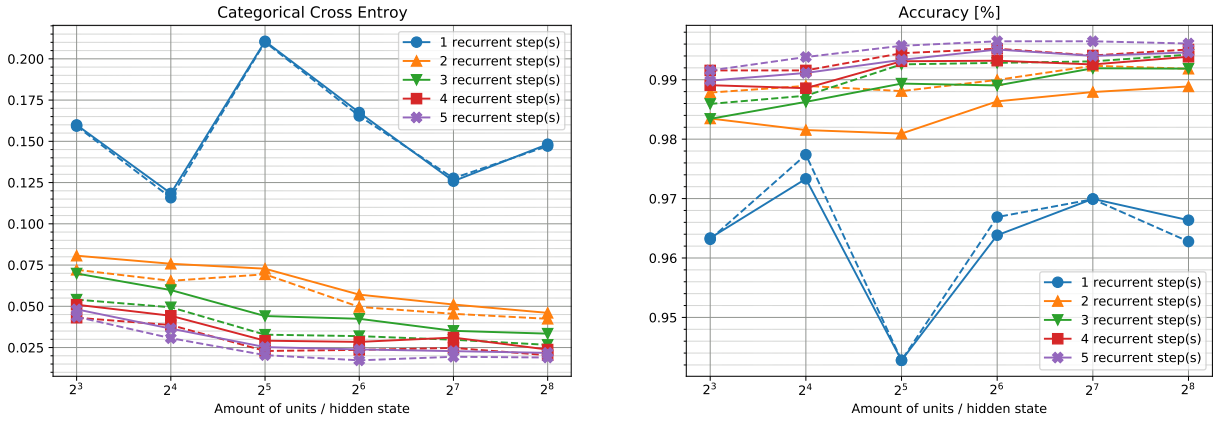


Figure 5.3: Training performances of LSTM RNNs which aim to predict the wake side with respect to a follower aircraft. Performances are represented for different amount of nodes in recurrent cells in function of the amount of previous time steps which are taken into account: plain lines (—) represent training performances while dashed lines (- - -) represent validation performances.

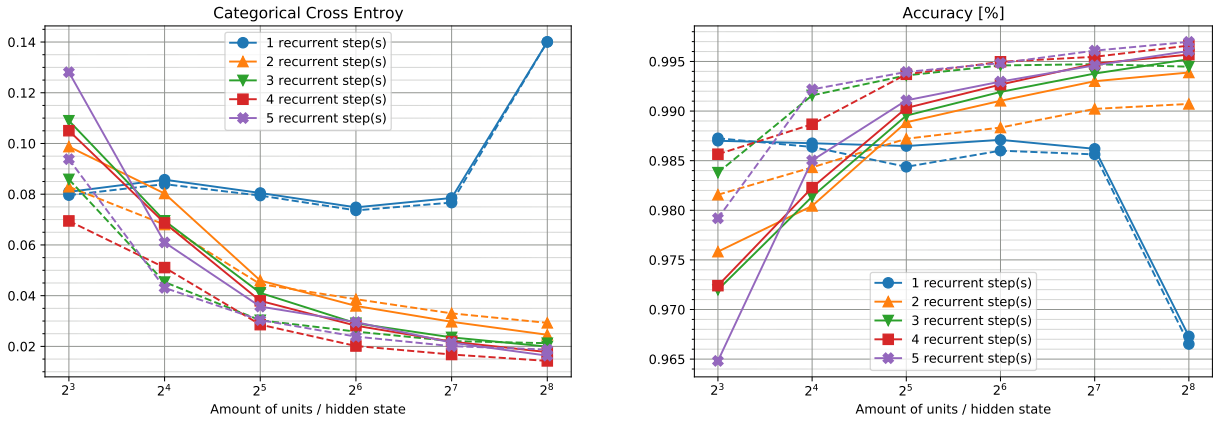


Figure 5.4: Training performances of GRU RNNs which aim to predict the wake side with respect to a follower aircraft. Performances are represented for different amount of nodes in recurrent cells in function of the amount of previous time steps which are taken into account: plain lines (—) represent training performances while dashed lines (- - -) represent validation performances.

In the end, one has to choose one model that shows better performances than others. Regarding previous comments, it is clear that we will focus our choice whether on LSTM or on GRU RNNs. Both showed similar results which are much better than classic feed-forward NNs and simple RNNs. Nevertheless since LSTMs have much more weights to compute, it seems preferable to work with GRU models. In that respect, we will even work while taking as much previous steps as possible into account and quite large amount of units too. Since red and purple lines on Figs. 5.4 are quite close to each other, we choose the less complex solution which is the red line. Concerning the amount of units, taking $N_{\text{units}} = 128$ seems quite reasonable.

Training results are similar for each model. Therefore we will only discuss about LSTM and GRU training performances for further problems. Nevertheless if the reader is interested in seeing other models' performances, they are presented in Appendix C.

5.1.2 Testing

As for verifying his assumptions after having chosen the right model, one wants to test it and see its accuracy. Therefore we represent on Fig. 5.6 the accuracy of the side predictor model according to the accuracy table of

Section 3.5. Since the total accuracy of the model is the percentage of true predictions, we verify that we obtain

$$\text{accuracy} = 50.26\% + 48.79\% = 99.05\% \quad (5.1)$$

which is largely above our first expectations (90%) and quite close to training performances (99.5%). Yet having 99% of true predictions bodes well for the results of further models.

As long as the accuracy gives a comparison on the hole range of data, one could be interested in the distribution of the accuracy with respect of the relative positions of the follower aircraft with the leader's wake. Given that, we discretized the oyz plane with the leader's wake located at the origin. Then given a certain relative position wrt. the wake, we may know each time the follower came accross and compute the accuracy. A contour plot on all the simulated box is presented on Fig. 5.7: we observe that the side prediction reaches 100% almost everywhere. That is to say there are few areas where it is actually decreasing. The first characteristic areas where the model is less performant is at the symmetry plane oxz with the wake center as the origin. These region could be more complicated for the model to predict because it is the intersection between the two prediction's classes. Therefore features' value may look similar to each other which is the reason why the prediction is made more complicated. Although the model lacks performance in these regions, we do not expect an aircraft to fly there. Let's indeed recall that there are intense downwash between the two vortices, which means that the aircraft uses much more thrust in aim to stabilize its flight. Therefore there are no interest for an aircraft to go there since formation flight aims to save energy and not waste it at the first opportunity.

On the other hand, we observe a second area where the accuracy decreases. The region starts from the vortices' center and has an hyperbolic shape. Although we observe the phenomenon portside wrt. the wake especially, one observes the same starboard yet it is weaker. As long as there is less data near the wake – because of the way we built the database – it is not surprising to observe worse performance there. Furthermore, the accuracy also seems to decrease with the distance because the intensity of the wake gets weaker and the influence becomes unnoticeable. By the way, this non-accuracy area is very similar to what observed DeVries et al. in their study on the wake's observability^[30]. Their work aimed to observe the wake parameters $[y_L, z_L, \Gamma_L]$ given the measurements of the differential pressure coefficients^[31]. Then they linearized their system and observed what is presented on Fig. 5.5 where we clearly observe the two hyperbolas but the observability seem to be quite good when we are close to the wake. All the same, one could conclude that these areas are zones where the observation of the wake is more difficult and having worse results there is not surprising. This being said, it is nonetheless quite interesting to observe that both strategy for tracking the wake of a leader aircraft lead their result to have the same global shape. We also point out that the diagonals are not identic because we did not use the same type of aircraft.

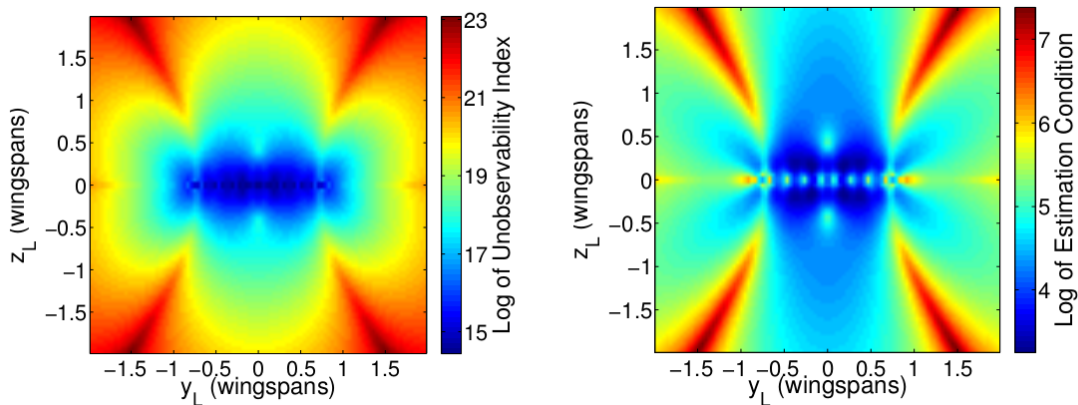


Figure 5.5: The log of the unobservability index (left) and estimation condition (right) over a range of leader aircraft relative position^[30].

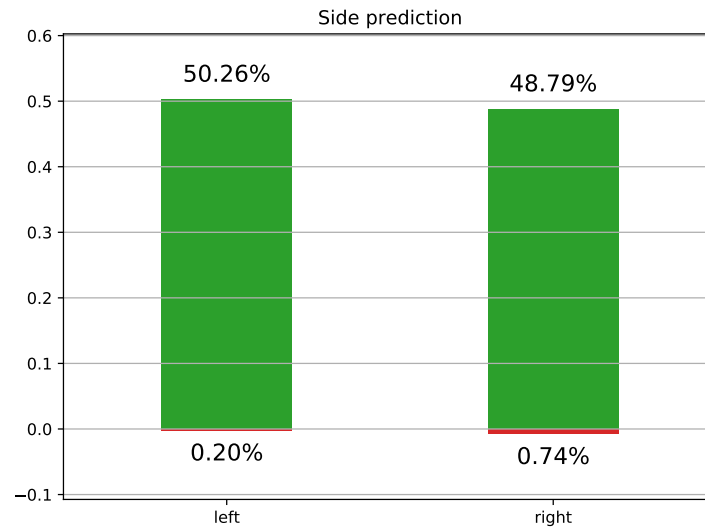


Figure 5.6: Testing accuracy on the prediction of the wake side wrt. a follower aircraft: the right prediction are in green and the false ones in red. Anoted percentages represent the amount of predictions holding in the corresponding column, the sum of each column's height being equal to 1.

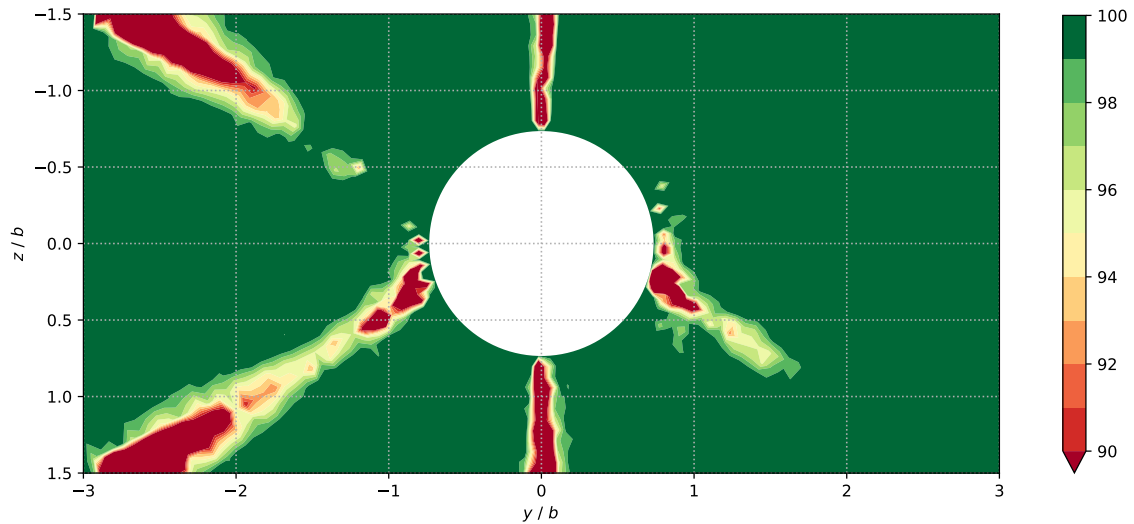


Figure 5.7: Distribution of the testing accuracy on the prediction of the wake side wrt. a follower aircraft on behalf of its position: green corresponds to 100% accuracy, yellow to 95% and red to 90% or below. The white circle at the center represents the wake of the leader and is an area where the follower could not go.

5.2 Orientation Approximation

5.2.1 Training

Performances of both LSTM and GRU RNNs are presented on Figs. 5.8 and Figs. 5.9, respectively. We discussed in Section 4.3 about the expectations that we have for models' performances. We also recall that we expected the MSE on the approximation of the orientation to be lower than $10 \text{ [deg]} = \pi/18 \text{ [rad]} \approx 0.1745 \text{ [rad]}$. It is verified on presented plots. What is more is that the MAE respects the same constraint: indeed, it is easy shown that

$$x^2 < x, \quad \forall x \in]0, 1[\Rightarrow \text{MAE} > \text{MSE} \quad (5.2)$$

for the current problem and results. Therefore the MAE is more compelling than the MSE.

Concerning the LSTM NNs, we observe as for the previous problem that the recurrency brings high performances to the results. Indeed, both the MSE and the MAE are decreasing while we are taking more previous time steps into account. We also observe that it tends to stabilize since red and purple lines are very close to each other. In addition to that, we observe that the errors does not vary so much with increasing amount of units, especially for the MSE. For these reasons we will keep a rather small amount of units and we will not go to much backward in time.

On the contrary, what one observes for GRU RNNs is more concerning. As expected, using the time continuous characteristic of a flight path is beneficial to performances. Yet, while we use more and more old time values, the model is not able to improve itself anymore. In fact, there is even no more solution and it blows up. This is very clear on Figs. 5.9 for red and purple lines. Moreover, although yellow and green lines are quite close to MAE values for LSTM NNs, their results on MSE are not which means that there are more errors above the unity.

Finally, it seems more rational to keep a LSTM models for further analysis because of their consistency and better results. On top of that, we will hold 3 previous time step in addition to the current one regarding the choice we have made for the side prediction model. The amount of units in the hidden state is later taken to be $N_{\text{units}} = 64$.

5.2.2 Testing

As for the side accuracy, one may start by giving a statistical approach to the error. This is given on Fig. 5.10 in the form of a probability density function. We directly observe that the error has a zero-mean which was of course expected. Next, one could try to fit this distribution with one of classic distribution functions. Since the gaussian curve is the most common, we first test it which is drawn in red. Its definition is written as

$$f(\varepsilon; 0, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp \left[-\frac{1}{2} \left(\frac{\varepsilon}{\sigma} \right)^2 \right] \quad (5.3)$$

Yet we observe that it does not fit the data so much. Indeed, the curve is too flat around the mean while it tends too fast to zero. We therefore test with the exponential distribution function which gives a better result ¹:

$$f(\varepsilon; 0, \sigma) = \frac{1}{2\sigma} \exp \left[-\frac{|\varepsilon|}{\sigma} \right] \quad (5.4)$$

Besides, the standard deviation value as been fixed to $\sigma = 1.2 \text{ [deg]} \approx 0.021 \text{ [rad]}$. Because of the definition of the exponential distribution function, one deduces that $\text{MAE} = \sigma = 0.021 \text{ [rad]}$ and that $\text{MSE} = 4.38 \times 10^{-4} \text{ [rad]}$ which both are in agreement to training results and much better to our previous expectations.

Although the most of data are gathered around zero, there are still a few laying near $\pm\pi \text{ [rad]}$. However the amount of these is so small that they could not be noticed on Fig. 5.10. One may still verify if these errors are induced by the wrong predictions of the first model. In that respect, we split right and left cases and verify for each one if the error is smaller to $\pi/2 \text{ [rad]}$ which means that the side prediction was correct. This is presented on Fig. 5.11 in terms of the accuracy; the result are similar to what we observed on Fig. 5.6 which supports that

¹The exponential distribution function normally is defined for $\varepsilon \geq 0$ but since our distribution has a zero-mean, we divided the main one by two and use $|\varepsilon|$ instead of ε .

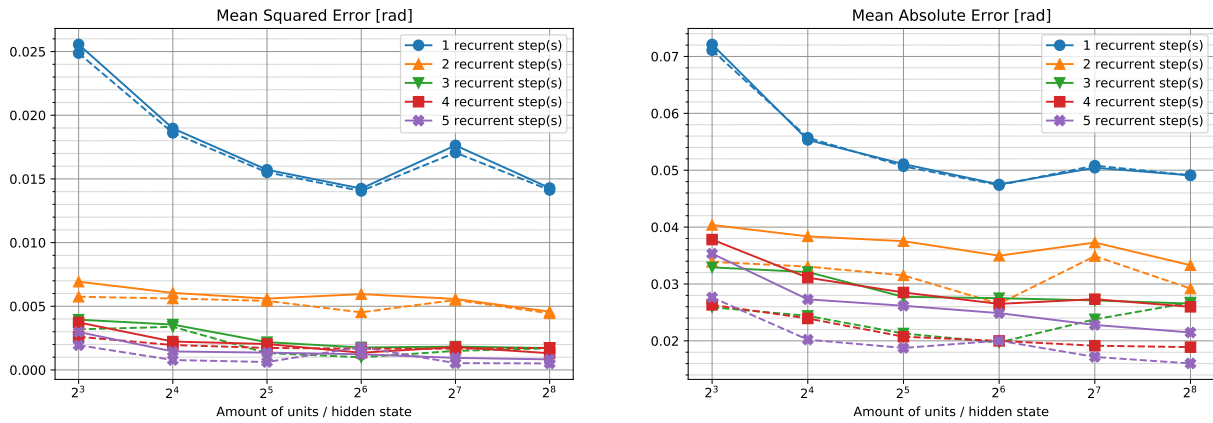


Figure 5.8: Training performance of LSTM RNNs which aim to approximate the orientation of the leader's wake wrt. the follower aircraft. Performances are represented for different amount of nodes in recurrent cells in function of the amount of previous time steps which are taken into account: plain lines (—) represent training performances while dashed lines (- - -) represent validation performances.

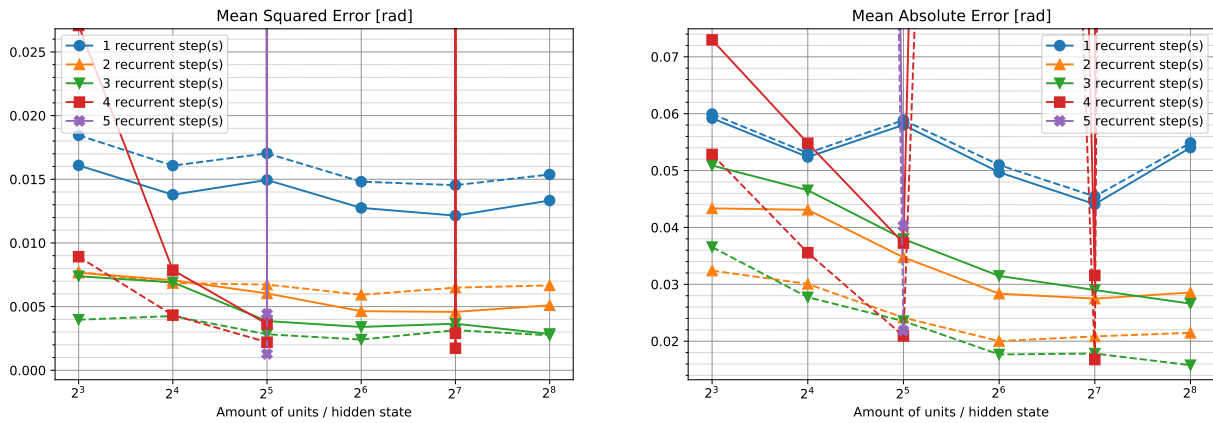


Figure 5.9: Training performance of GRU RNNs which aim to approximate the orientation of the leader's wake wrt. the follower aircraft. Performances are represented for different amount of nodes in recurrent cells in function of the amount of previous time steps which are taken into account: plain lines (—) represent training performances while dashed lines (- - -) represent validation performances.

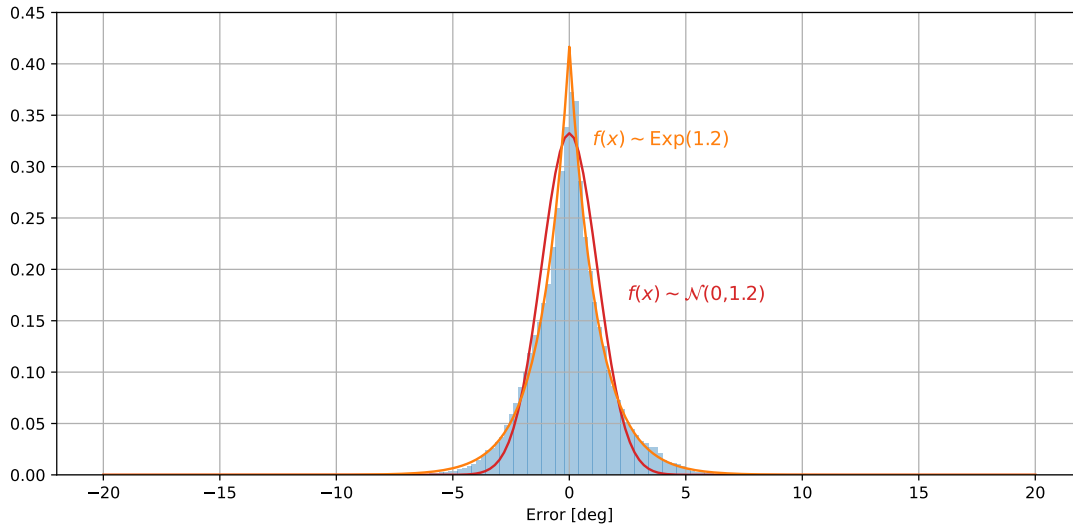


Figure 5.10: Error distribution of the approximation of the wake orientation wrt. the follower aircraft. Displayed errors lay in the range $[-20; 20]$ [deg]. There are of course at higher values but actually so few that they cannot be observable.

bad predictions on the first model brings bad approximations for the second one.

In addition to that, we represented on Fig. 5.12 the distribution of the MAE in function of the relative position of the follower aircraft wrt. the wake of the leader. As long as the approximation of the orientation is directly related to the prediction of the side, we observe here the same global result as on Fig. 5.7. Indeed, we observe the same two types of areas where the error is getting worse. Concerning the intersection between the right and left predictions (first model), it was not surprising to observe an increase of the error in this area. However, it is somewhat more interesting to also observe it here. Indeed, we even observe that the error is going up to 40 degrees which becomes quite important regarding the range of expected values!

Otherwise we again observe the hyperbolic shape on both side of the wake, yet less noticeable starboard. Moreover, the error also seems to increase when the aircraft comes very close to the wingtip vortices and when the distance gets higher. The observation gets even better than previously since the MAE is higher than $\pi/2$ [rad]. Besides, the most of the area gives quite high performances with a MAE laying between 0 and 5 degrees.

5.3 Approach Prediction

5.3.1 Training

Respective training results of LSTM and GRU RNNs are presented on Figs. 5.13 and Figs. 5.14. We recall that we expected an accuracy above 90% in aim to have reliable results when coupling with other models. This constraint has been largely satisfied regarding the 98% accuracy of several models. We still observe the positive impact of recurrency in the models which allows us to restrict our analysis to LSTM and GRU networks only.

By contrast to other problems, we observe that whether the network, using more than 1 previous time step does not improve the results, and all corresponding lines on the figures gather in real mess. Concerning results about LSTM RNNs, we still do not observe improvement while the amount of nodes in the cell is increasing. Conversely, we observe spurious results without any stability. Even if they are close to each other, one cannot say much about them because there is no consistency.

On the other hand, we observe much more stable results for GRU performances, especially when more than 2 previous time steps are taken into account – it means lines in green, red and purple. It would mean that for gaining some consistency, the models need to gain in complexity. In fact, if they are too simple in regards to a

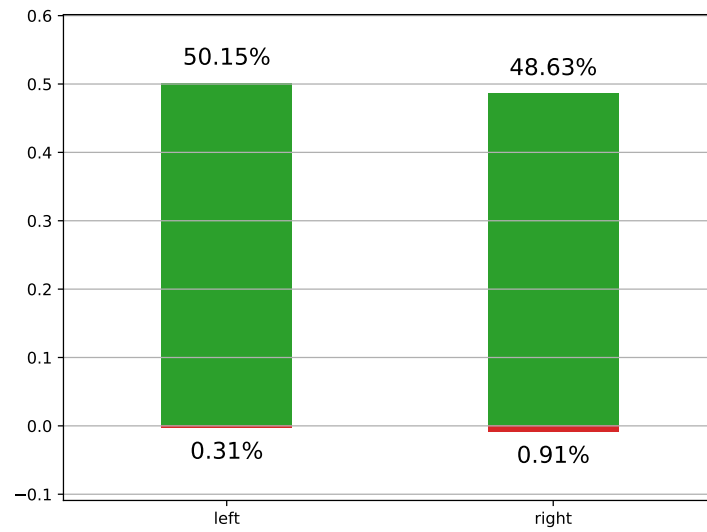


Figure 5.11: Accuracy of the prediction of the wake side wrt. the follower aircraft based on its approximation error of the orientation: the right prediction are in green and the false ones in red. Anoted percentages represent the amount of predictions holding in the corresponding column, the sum of all being equal to 1.

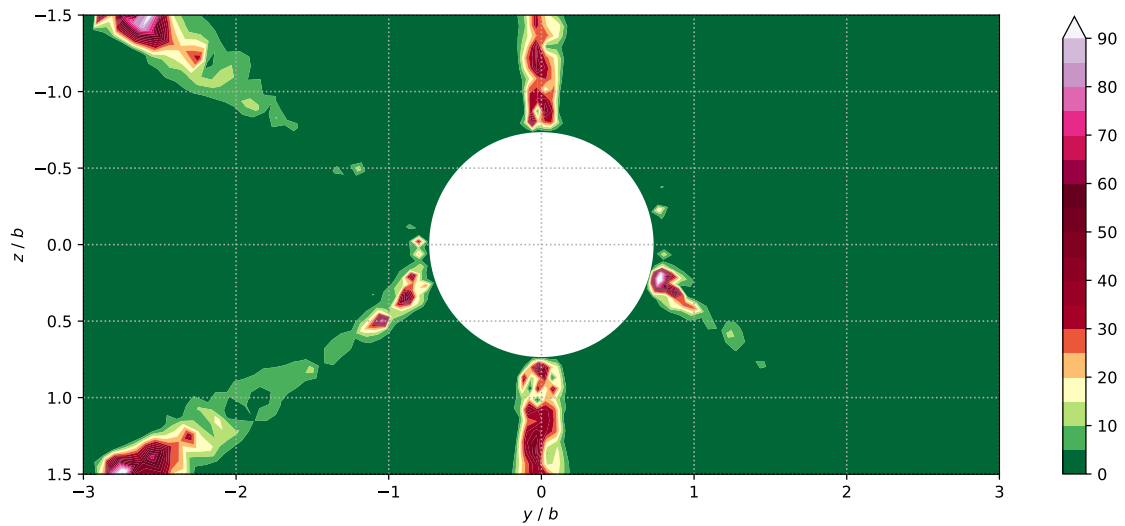


Figure 5.12: Distribution of the mean absolute error [deg] on the approximation of the wake orientation wrt. a follower aircraft on behalf of the relative position between them – the colorbar on the side presents the error value [deg] relatively to the figure. The white circle at the center represents the wake of the leader and is an area where the follower could not go.

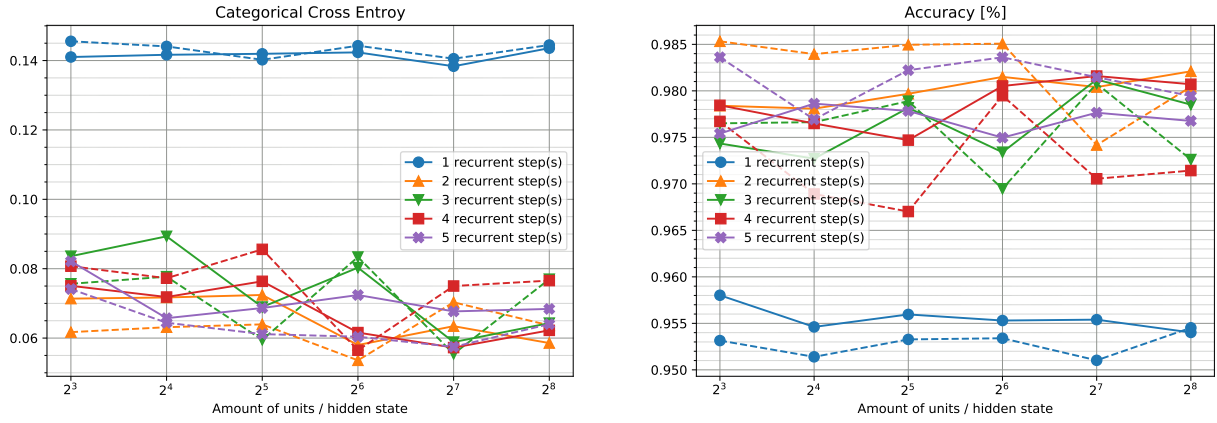


Figure 5.13: Training performances of LSTM RNNs which aim to predict if the follower aircraft come closer to the leader, or not. Performances are represented for different amount of nodes in recurrent cells in function of the amount of previous time steps which are taken into account: plain lines (—) represent training performances while dashed lines (- - -) represent validation performances.

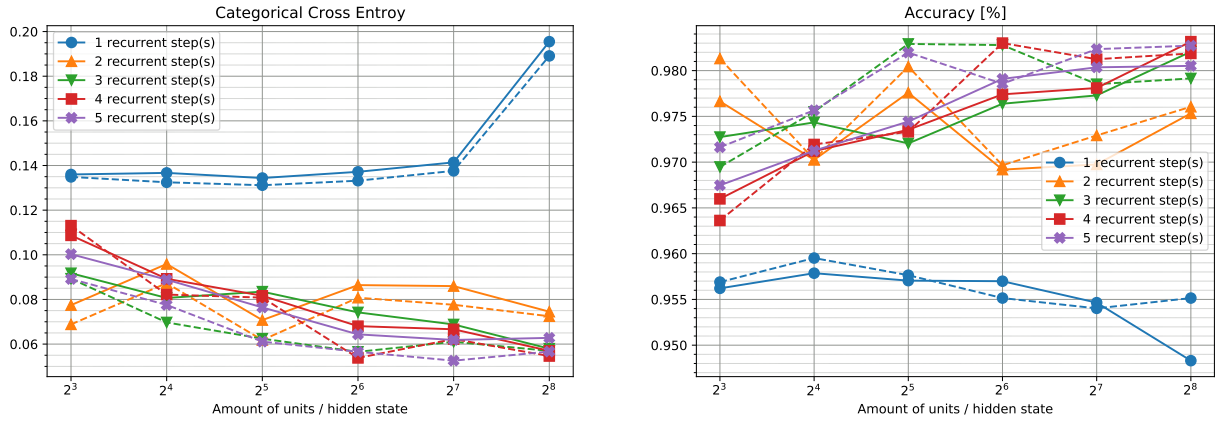


Figure 5.14: Training performances of GRU RNNs which aim to predict if the follower aircraft come closer to the leader, or not. Performances are represented for different amount of nodes in recurrent cells in function of the amount of previous time steps which are taken into account: plain lines (—) represent training performances while dashed lines (- - -) represent validation performances.

complex problem, they might actually not be able to give accurate results. Of course the question is not asked for our purpose since accuracy is above 98% but the decision about a best model will still be impacted. With regards to the discussion, GRU models seem to be more efficient for the current approach problem. For the sake of complexity, we will take a respectable amount of previous time steps into account. Therefore 5 recurrent steps seems to be reasonable. Finally, we choose $N_{\text{units}} = 128$.

5.3.2 Testing

As we did for the first model, we present the accuracy on the approach prediction on Fig. 5.15. We observe here that the results are largely better than first expectations which was 81%. Yet, we deduce from the figure that accuracy reaches

$$\text{accuracy} = 48.38\% + 46.22\% = 94.6\% \quad (5.5)$$

which is less than training performances where the accuracy was around 98%. Nonetheless, this last model is highly dependent to the two previous ones which might explain this difference in the results. Since their density function are not correlated – the two models have been trained separately and have been gathered for the testing

phase only – one deduces that

$$P_{\text{testing, approach}} = P_{\text{side}} \times P_{\text{training, approach}} = 99\% \times 98\% = 97\% \quad (5.6)$$

which does not fully explain the 94.6% that we observe. But maybe tackling this problem through the side prediction is not the right way because it does not take all the error into account. Indeed, the orientation of the wake is the output of the second model but also an input of the approach prediction. Because of that, only relying on the side prediction may be not accurate at all. This is the reason why we compare the accuracy of the approach prediction with the error made on the approximation of the orientation. Because of the high variation in the results, we prefer to smooth them by using the coefficients of Savitzky-Golay filter given in the table hereunder^[32]

Window Size	-4	-3	-2	-1	0	1	2	3	4	Normalization
5			-3	12	17	12	-3			35
7		-2	3	6	7	6	3	-2		21
9	-21	14	39	54	59	54	39	14	-21	231

The coefficients are given in function of the relative absisce of the studied curve. The smoothed function then reads following the relation

$$Y_\varepsilon = \sum_{i=\frac{1-m}{2}}^{\frac{m-1}{2}} C_i y_{\varepsilon+i}, \quad \frac{m-1}{2} \leq j \leq n - \frac{m-1}{2} \quad (5.7)$$

where C_i are the ratio between SG's coefficients and the normalization factor and ε is the orientation's error. Besides that, we are also interested in the comparison of the latter accuracy with the amount of times errors have been made. In other words, the more a given error has been generated, the more the corresponding accuracy will have weight on the total accuracy. Nonetheless we observed on Fig. 5.10 that most of errors are distributed between 0 and 5 degrees and that the probability of having an error outside the range $[-5, 5]$ degrees is nearly zero. Therefore we will use the log of the amount of sample. Finally, we come up drawing a bar plot where bars' height at orientation error ε is given by

$$y_\varepsilon = (-1) \times \text{accuracy}_\varepsilon \times \log \left(\frac{N_\varepsilon}{N_{\text{total}}} \right) \quad (5.8)$$

It means that the bigger a bar is, the less the error has been made. Yet where N_ε 's are constant for a large range of errors – it means for $|\varepsilon| \in [20; 150]$ degrees, approximately – then the accuracy takes the lead and makes the difference. These two plots are presented on Fig. 5.16 where the accuracy has been smoothed over a window of 9 samples. We first observe that at small errors, the accuracy climbs over 95%, even approaches the accuracy rate we observed during training and is quite constant. However it decreases fast regarding the 85% accuracy for 10 degrees errors. It even goes below 80% after the absolute error made on the orientation has passed 80 degrees. Then it gets worse by decaying below 50% for absolute error higher than 150 degrees. Yet it might seem normal to get a symmetry past 90% because of the way χ has been defined. Since we do not observe that symmetry, one may expect that both $\Delta\dot{z}$ and $\Delta\dot{x}$ have an influence and give valuable information on the variation in vortices' intensity. On the other, the periodicity of the scheme is verified since the accuracy at $\varepsilon = \pi$ radians equals that accuracy at $\varepsilon = -\pi$ radians.

Finally the distribution of the approach prediction in function of the relative position of the follower wrt. the wake is presented on Fig. 5.17. Yet we observe here that the accuracy is quite constant and we do not observe the same errors' areas as before. Nonetheless one may still observe that the accuracy is slightly decreasing with the distance but not enough for more detailed explanations. At least, one deduces that the prediction on the approach is not correlated with the orientation, neither it is related to the observation of the wake. In fact, this would even mean that both $\Delta\dot{z}$ and $\Delta\dot{\phi}$ model's inputs have no big impact on the prediction.

5.4 Discussion

This chapter was focused on training and testing performances of three models which were better than first expectations and very similar to observations of other works. However it seems important to get some perspective

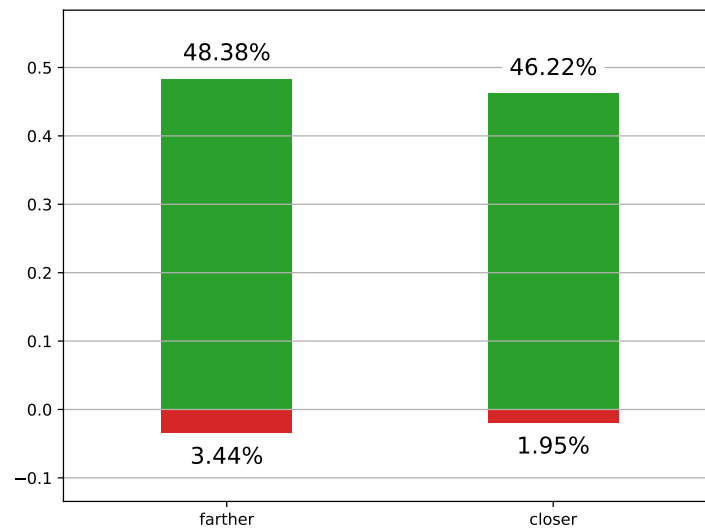


Figure 5.15: Testing accuracy on the prediction of the wake side wrt. a follower aircraft: the right prediction are in green and the false ones in red. Anoted percentages represent the amount of predictions holding in the corresponding column, the sum of each column's height being equal to 1.

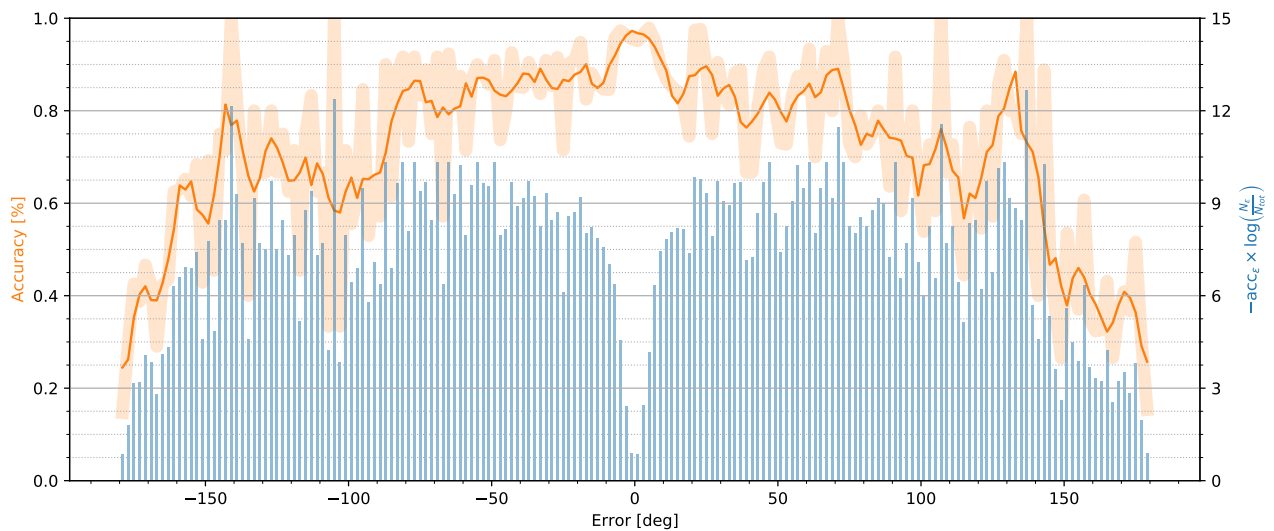


Figure 5.16: Accuracy on the approach error as a function of the error made on the approximation of the orientation (orange) compared with the same accuracy multiplied by the log ratio of the amount of data corresponding to the error (blue).

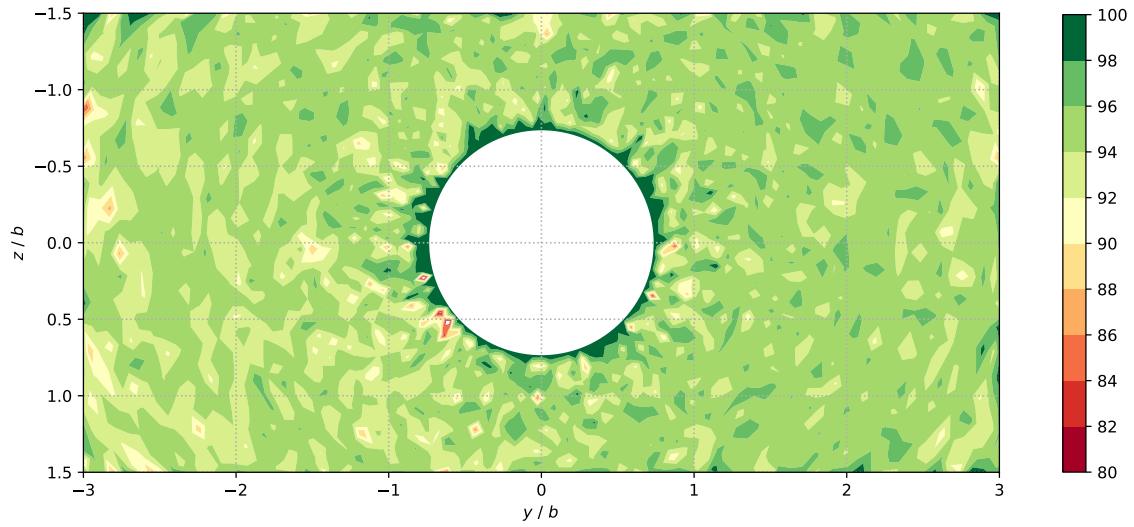


Figure 5.17: Distribution of the testing accuracy on the prediction of the wake side wrt. a follower aircraft on behalf of its position: green corresponds to 100% accuracy, yellow to 95% and red to 90% or below. The white circle at the center represents the wake of the leader and is an area where the follower could not go.

and pick important information that could be interesting for future works. First, we observed that RNNs increased a lot the performances and therefore they have been preferred to feed-forward NNs. On top of that, GRU RNNs showed better results for classification problems while they were blowing up on the regression one, conversely to LSTM RNNs. While performances were still increasing with the amount of previous time steps considered, we also observed that it tends to stabilize past 3 to 5 recurrence steps. Increasing the amount of units in the hidden state was also giving big improvements but stabilizes them past 128 units approximately.

Furthermore, we noticed that testing performances were very close to training ones, and much better than any expectations. However we observed that the error on the side prediction and the orientation of the wake were distributed the same way as the unobservability studied by DeVries et al. in their paper^[30]. Concerning the approach prediction, we observed that its distribution in function of relative positions was much more constant. As long as it is directly dependent on the approximation of the orientation, it is affected by the error of this approximation. In that respect we observe that performances decreased very fast, particularly for absolute error higher than 10 degrees. Yet, since the most of the errors are laying between -5 and 5 degrees, the global performances of the model remain quite good. Although these regions of unobservability are regions where the performances are getting worse, the fact that we are aware of that is valuable information and could be used. Remember when we talked in Section 4.2 about a discretization of the space around a wake into several zones. Now that we know how the models are performing depending on the relative position of the follower aircraft, one can now split the (oyz) plane into three regions: the first one is the zone inside the hyperbolas on both sides of the wake, the second one is the hyperbolas themselves where the observability of the wake decreases and the third one is the regions where induced speeds of the wake are downward. In that respect, an aircraft is able to predict if its knowledge about the wake is trustable or not.

Besides, there are some points that one must be aware of. Indeed, the way that the database has been built is simplified which may not perfectly correspond to the reality. Global atmosphere turbulence has not been considered and LLs theory takes assumptions which biases the point of view. On top of that, all measures are “true”: this means that sensors aboard of the aircraft do not induce any noise in measurements. Therefore all the database is perfect and smooth which eases a lot the training and might considerably improve the performances. Yet, in reality the measurements are distributed around the true value according to a normal distribution of a given standard deviation relative to sensors specifications. Given that, one rationally expects that errors’ areas will spread more and local performances will globally decrease.

In this work, we verified the ability of NNs to characterize the wake of an aircraft based on onboard measurements of a follower aircraft. A key point in supervised learning is the complexity of the studied models, in their internal structure as well as in the selected features to meet the objectives. In that respect, we went through a deep analysis of all available measures and found out that there are actually 8 which are directly affected by the environment induced speeds. These were used as the inputs of the different models as well as some other useful measures depending on the purpose.

Next, we compared the training performances of several models from classic feed-forward NNs up to RNNs. We observed that RNNs were actually outperforming feed-forward NNs which was of course expected since the flight path of an aircraft is continuous in time. In addition to that, GRU RNNs showed better performances on classification problems while they were blowing up on the regression problem. In comparison, LSTM RNNs did not blow up for any model and their performances were very stable. On top of that, we observe a continuous increase of performances with the amount of recurrent steps and the amount of units in the hidden state. Finally, we choose to keep GRU RNNs for the two classification problems and a LSTM RNN for the regression one. For the sake of complexity and for the computational memory, we chose to keep 4 to 5 recurrent steps and to build models with 64 or 128 units in the hidden state.

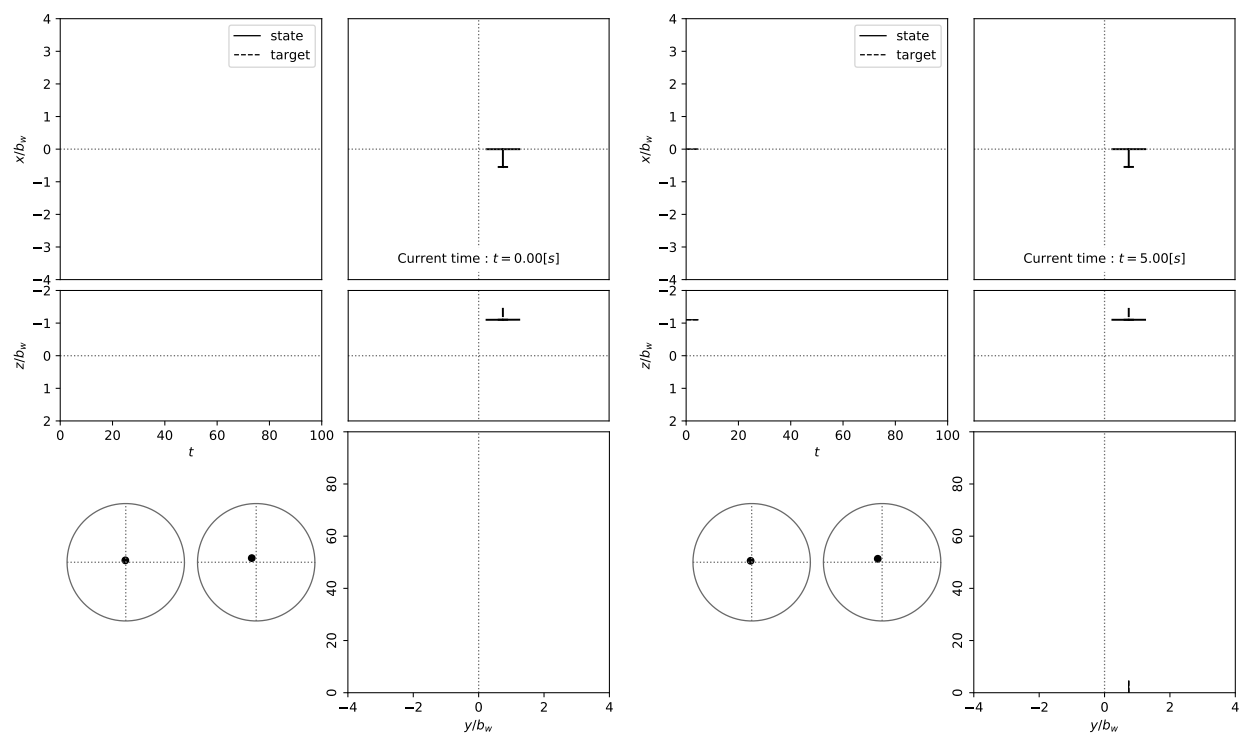
Besides, testing performances are close to training ones and much better than any expectations. Furthermore, the particular shape of results are very similar to what observed DeVries et al. in their study of the observability of aircraft's wakes based on pressure differences in the spanwise direction. Indeed, it is clear that models are lacking performance in regions where the wake is less observable. Although it is obvious for the prediction of the side and the approximation of the orientation, the precision on the approach seems much more stable with respect to relative positions. On top of that, we also observe a lack in performance at the wake's plane of symmetry.

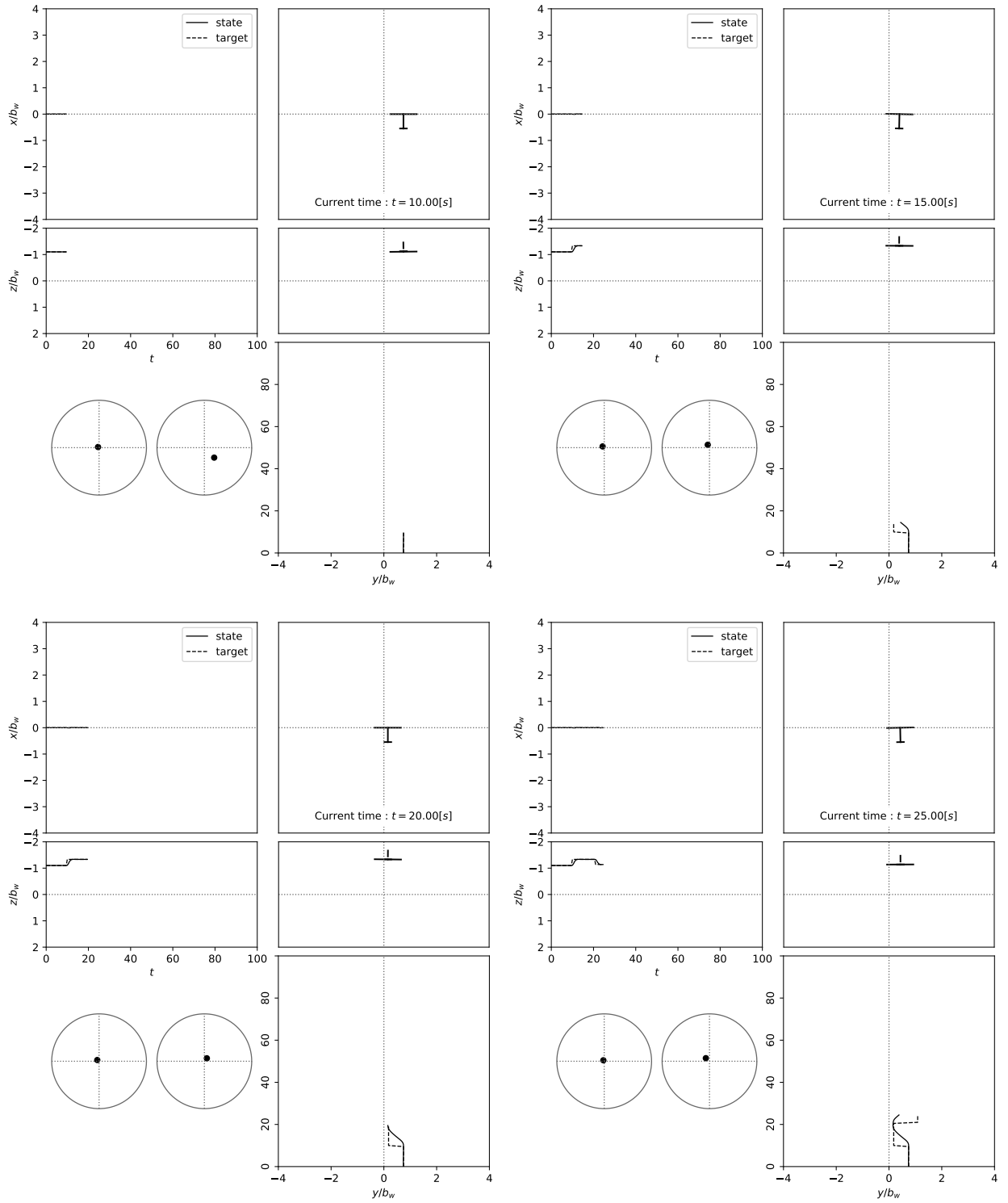
Although results were very promising, one must be careful by not trusting them entirely. Indeed, the way the models have been built is simplified and measures on which we have worked may be not reflect the reality as well as one could expect. Furthermore, there is no noise in the measures which makes them perfect and smooth. This helps a lot the models during the training phase and might give bad results in real life experiments.

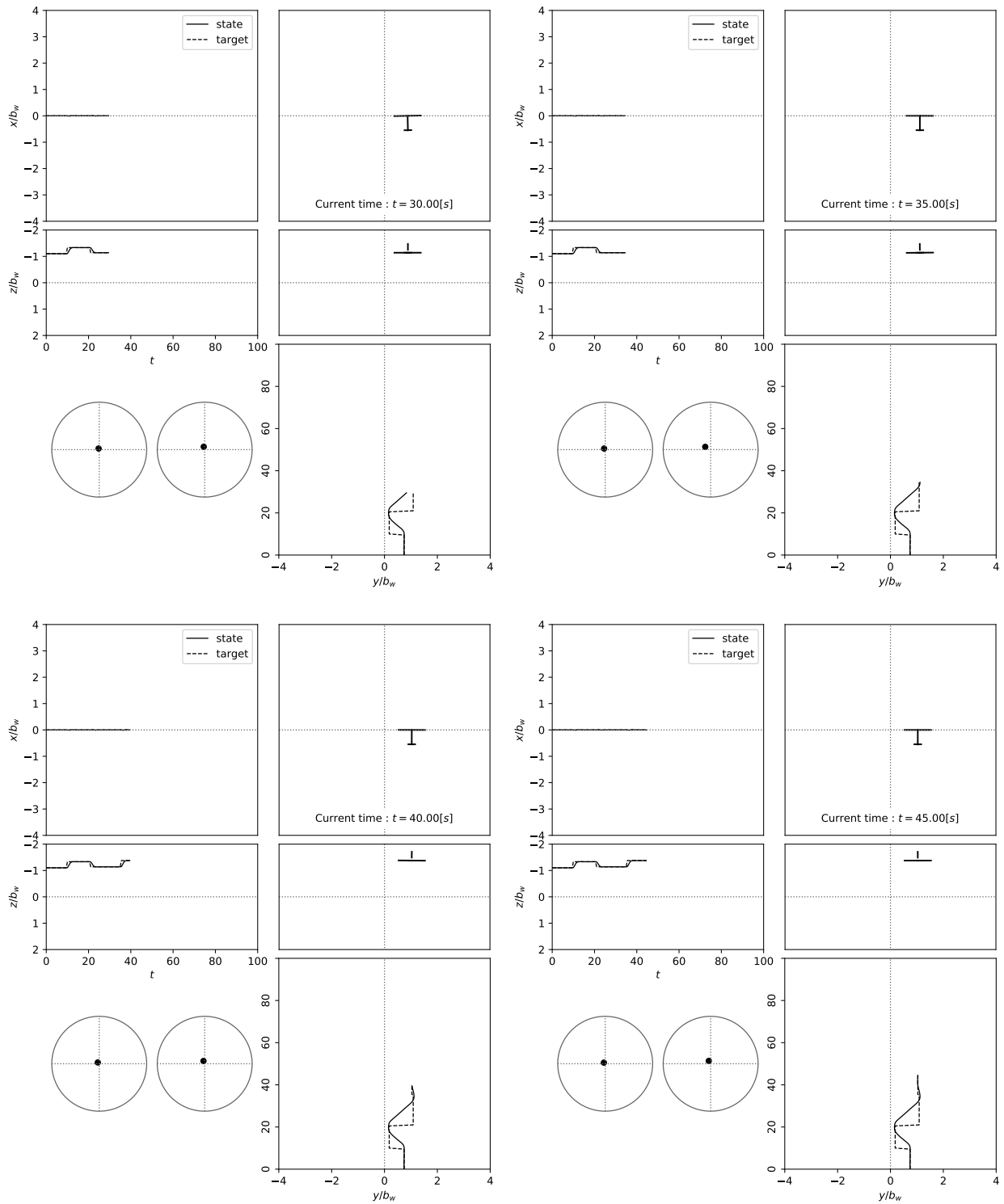
The above results are useful since they give some valuable knowledge about the environment of a flying aircraft, and help it to localize efficiently the wake of a leader. Nevertheless, the purpose of the study being the formation flight, one is interested in controlling the aircraft and keeping it at the sweet spot. Since aircraft wakes remain very dangerous to any encountering aircraft, we prefer not to get involve into the calibration of control commands. In that respect, we will rather build a tool which is able to determine optimal control commands and reinforcement learning is considered as a possible continuity of the present work.

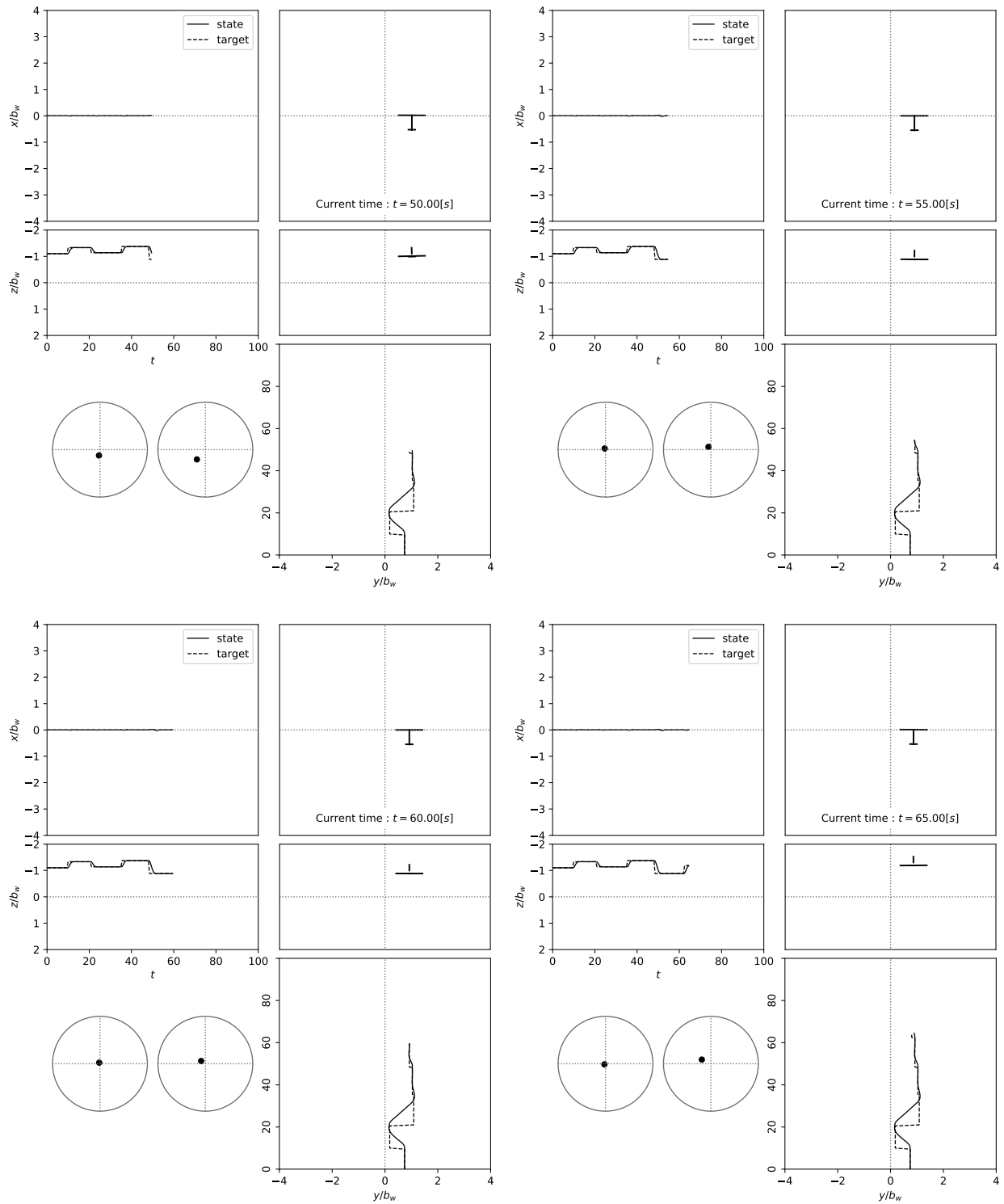
APPENDIX A

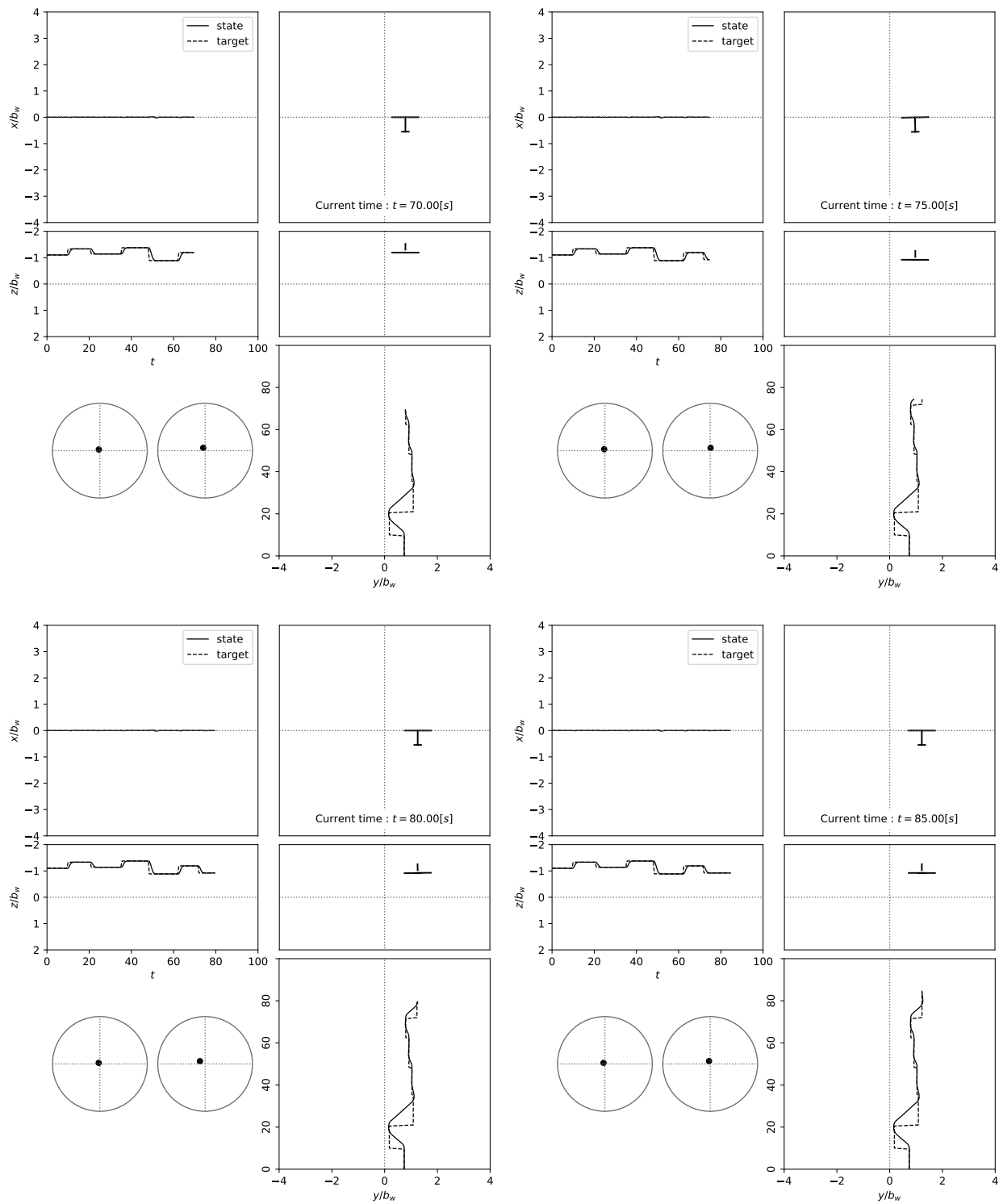
ANIMATION OF A RANDOM TARGET POLICY

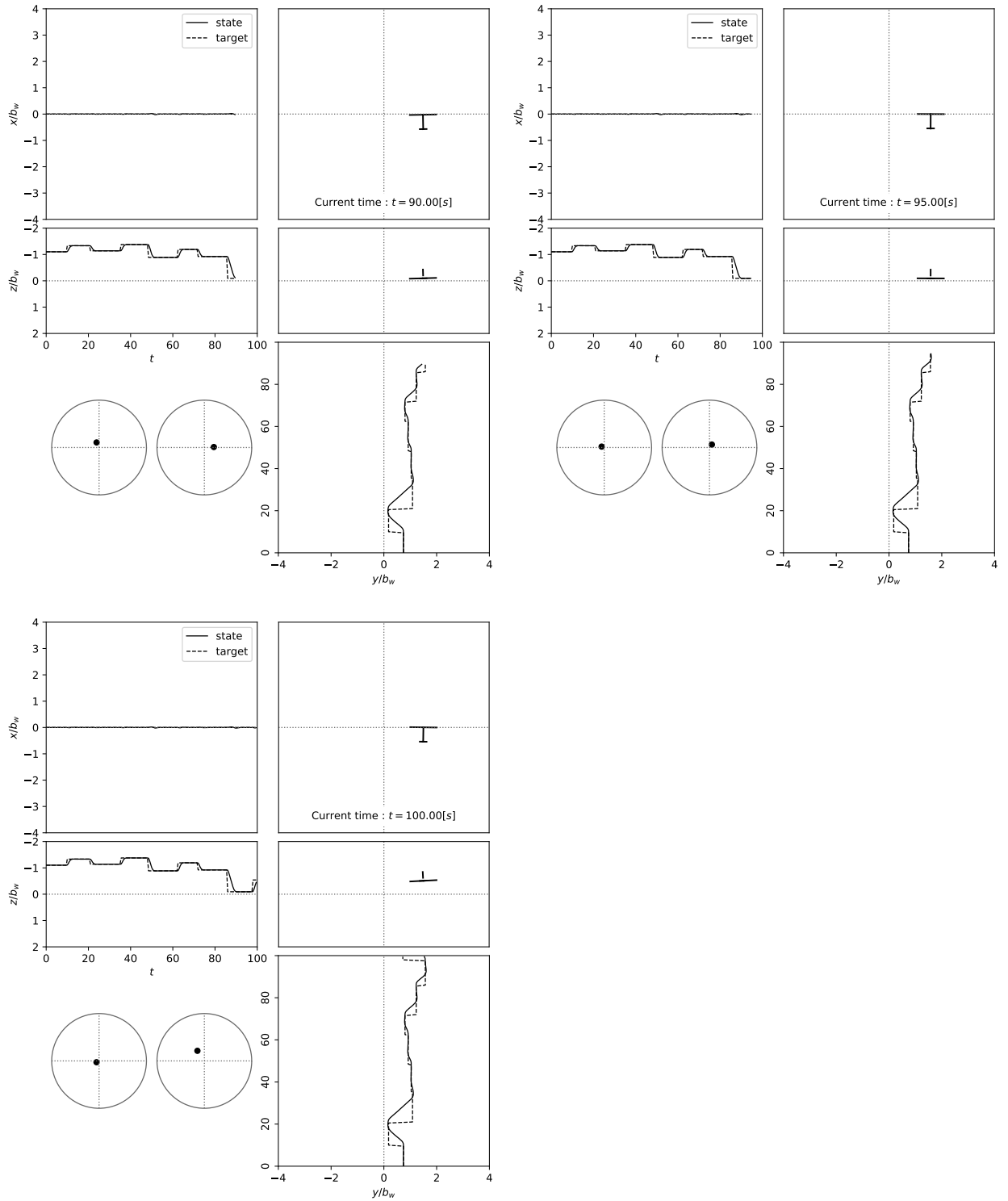












APPENDIX B

LIFTING LINE THEORY

In aim to build the mathematical model for lifting lines, we first assume that the lift is function of the spanwise direction only and because of the structure, there is no circulation at wingtips which involves that $\Gamma(-b/2) = \Gamma(b/2) = 0$. Since the total lift reads

$$L = \rho V_\infty \Gamma_0 b \quad (\text{B.1})$$

one may deduce the lift at spanwise location y as

$$l(y) = \rho V_\infty \Gamma(y). \quad (\text{B.2})$$

In addition to that, the continuity of the vortex tube imposes that we define a vortex sheet as

$$\Gamma(y + dy) = \Gamma(y) + \frac{d\Gamma}{dy}(y) dy \quad (\text{B.3})$$

where the second term of right hand side part is the circulation of elementary vortex tubes. Therefore the circulation per unit length of the vortex sheet is

$$\gamma(y) = -\frac{d\Gamma}{dy}(y) \quad (\text{B.4})$$

Then, one integrate the lift along the spanwise direction in aim to obtain the total lift

$$L = \int_{-b/2}^{b/2} l(y) dy = \rho V_\infty \int_{-b/2}^{b/2} \Gamma(y) dy \quad (\text{B.5})$$

As depicted on Fig. 2.2, the true velocity as feeled by the airfoil has not the same direction as the stream velocity. In fact, the air slightly modifies its direction while it is approaching to the wing. This is represented by the vertical downwash velocity $w\mathbf{e}_z$ for $w \ll V_\infty$. The local downwash is expressed in term of the circulation as

$$dw(y) = -\frac{1}{4\pi} \frac{1}{y - y'} \frac{d\Gamma}{dy'}(y') dy' \quad (\text{B.6})$$

which integration along the span of the wing gives the total downwash at a specific location y :

$$w(y) = -\frac{1}{4\pi} \int_{-b/2}^{b/2} \frac{1}{y - y'} \frac{d\Gamma}{dy'}(y') dy'. \quad (\text{B.7})$$

Consequently, the angle of attack is decreased of the angle of downwash ϵ which result is the effective angle of attack

$$\alpha_e(y) = \alpha(y) - \epsilon(y) = \alpha(y) - \left(\frac{-w}{V_\infty} \right). \quad (\text{B.8})$$

Observe that α is function of the spanwise location y : this is actually the case if the wing is twisted.

In addition to that, the force that the wing is subjected to is perpendicular to the true air speed as feeld by the wing. Therefore the total force is splitted into one horizontal component d_i and a vertical one l . These read

$$l(y) = l(y) \cos(\epsilon(y)) \approx l(y) \quad (\text{B.9})$$

$$d_i(y) = l(y) \sin(\epsilon(y)) \approx l(y)\epsilon(y) \quad (\text{B.10})$$

because $\epsilon(y)$ is taken to be small. The vertical force is already known and represent the lift component while d_i is the drag component. One integrates it along the spanwise and obtains

$$D_i = \int_{-b/2}^{b/2} d_i(y) dy = \int_{-b/2}^{b/2} l(y)\epsilon(y) dy \quad (\text{B.11})$$

which is called the induced drag. In other words, it is the drag induced by the vortex sheet due to lift.

On the other hand, the wing's profile at y also has its own characteristics, namely the lift coefficient

$$C_l(y) = \frac{l(y)}{\frac{1}{2}\rho V_\infty^2 c(y)} = a_0 \alpha_e(y) \quad (\text{B.12})$$

where $c(y)$ is the chord length of the profile and $a_0 = 2\pi$ is the derivation of the lift coefficient with respect to the angle of attack. One isolates α_e from last equation and develops $l(y)$:

$$\alpha_e(y) = \frac{2\rho U_\infty \Gamma(y)}{\rho U_\infty^2 c(y) a_0} = \frac{2}{V_\infty} \frac{\Gamma(y)}{a_0 c(y)}. \quad (\text{B.13})$$

By gathering Eq. (B.8) and Eq. (B.13), one finally obtains an expression for the circulation

$$\Gamma(y) = \frac{1}{2} a_0 c(y) V_\infty \left[\alpha(y) - \frac{1}{4\pi} \frac{1}{V_\infty} \int_{-b/2}^{b/2} \frac{1}{y - y'} \frac{d\Gamma}{dy'}(y') dy' \right] \quad (\text{B.14})$$

which is known as the compatibility equations and has been developped by Prandtl in 1918^[16]. It allows to determine the circulation Γ along wing span, from which we deduce all flight characteristics: L , D_i , $\alpha_e(y)$, etc. This equation is solved by using the change of variable

$$y = \frac{b}{2} \cos \theta \quad (\text{B.15})$$

and by assuming a symmetrical circulation along the wingspan:

$$\Gamma(\theta) = b V_\infty \sum_{n \text{ odd}} A_n \sin(n\theta). \quad (\text{B.16})$$

Therefore, Eq. (B.14) becomes

$$\Gamma(\theta) = \frac{1}{2} a_0 c(\theta) V_\infty \left[\alpha(\theta) - \frac{1}{4\pi} \frac{1}{V_\infty} \int_0^\pi \frac{1}{\frac{b}{2}(\cos \theta - \cos \theta')} \frac{d\Gamma}{d\theta'}(\theta') d\theta' \right] \quad (\text{B.17})$$

in which we inject Eq. (B.16):

$$b \sum_{n \text{ odd}} A_n \sin(n\theta) = \frac{1}{2} a_0 c(\theta) \left[\alpha(\theta) - \frac{1}{2\pi} \sum_{n \text{ odd}} n A_n \underbrace{\int_0^\pi \frac{\cos(n\theta')}{(\cos \theta - \cos \theta')} d\theta'}_{\pi \frac{\sin(n\theta)}{\sin \theta}} \right] \quad (\text{B.18})$$

It finally gives the linear system

$$\sum_{n \text{ odd}} A_n \left[1 + \frac{a_0}{4} \frac{c(\theta)}{b} \frac{n}{\sin \theta} \right] \sin(n\theta) = \frac{a_0}{2} \frac{c(\theta)}{b} \alpha(\theta). \quad (\text{B.19})$$

where only unknowns are coefficients A_n . These developments have been taken from the course given by G. Winckelmans about aerodynamic of external flows^[33].

APPENDIX C

TRAINING PERFORMANCES

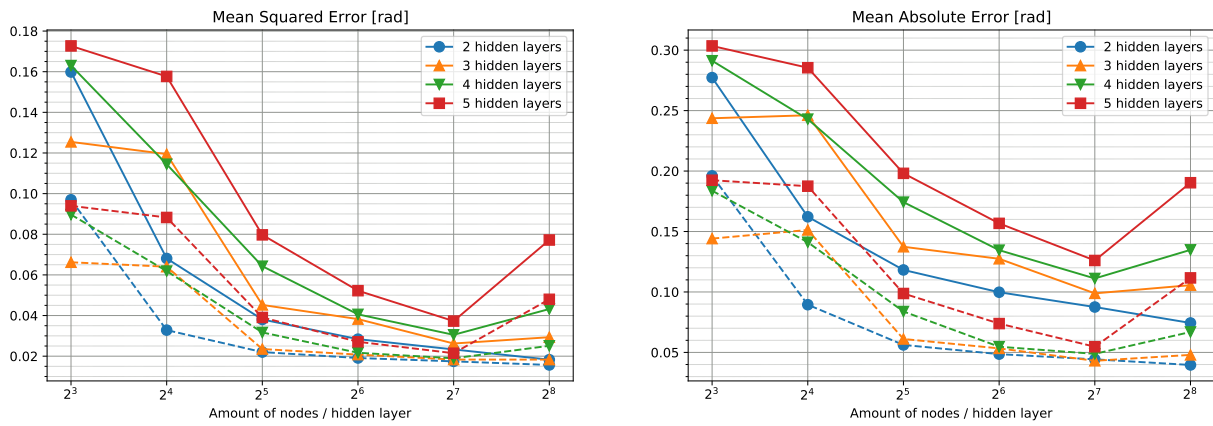


Figure C.1: Training performance of feed-forward NNs which aim to approximate the orientation of the leader's wake wrt. the follower aircraft. Performances are represented for different amount of hidden layers at certain amount of nodes per hidden layer: plain lines (—) represent training performances while dashed lines (- - -) represent validation performances.

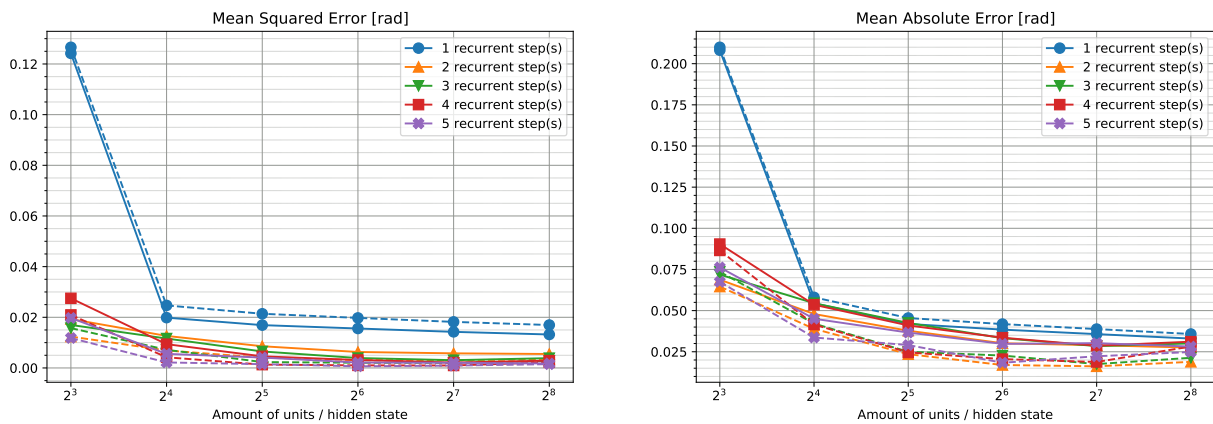


Figure C.2: Training performance of simple RNNs which aim to approximate the orientation of the leader's wake wrt. the follower aircraft. Performances are represented for different amount of nodes in recurrent cells in function of the amount of previous time steps which are taken into account: plain lines (—) represent training performances while dashed lines (- - -) represent validation performances.

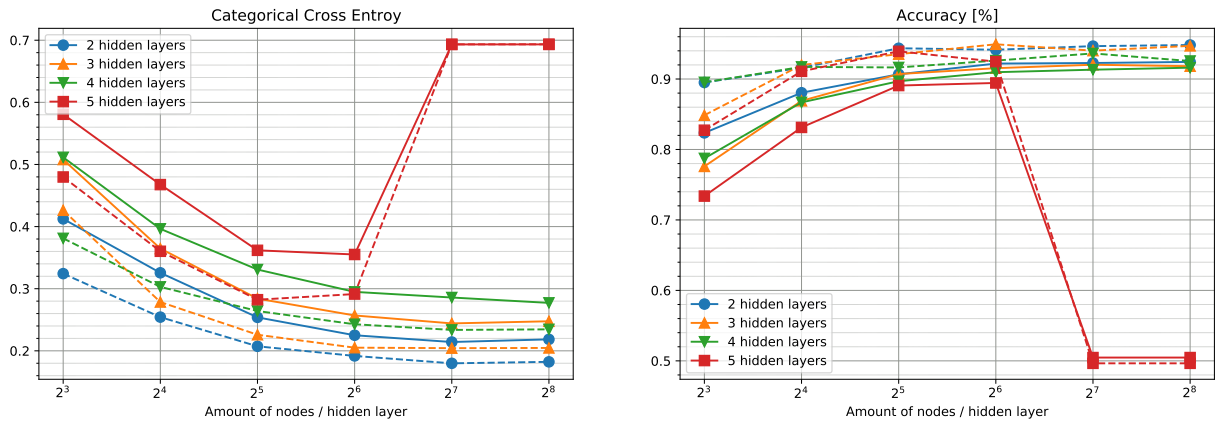


Figure C.3: Training performances of feed-forward NNs which aim to predict if the follower aircraft come closer to the leader, or not. Performances are represented for different amount of hidden layers at certain amount of nodes per hidden layer: plain lines (—) represent training performances while dashed lines (- - -) represent validation performances.

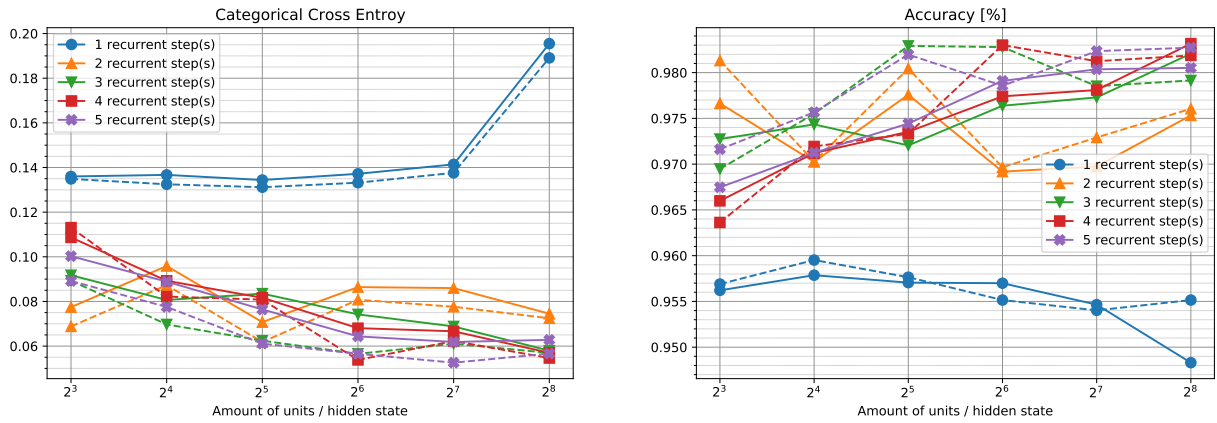
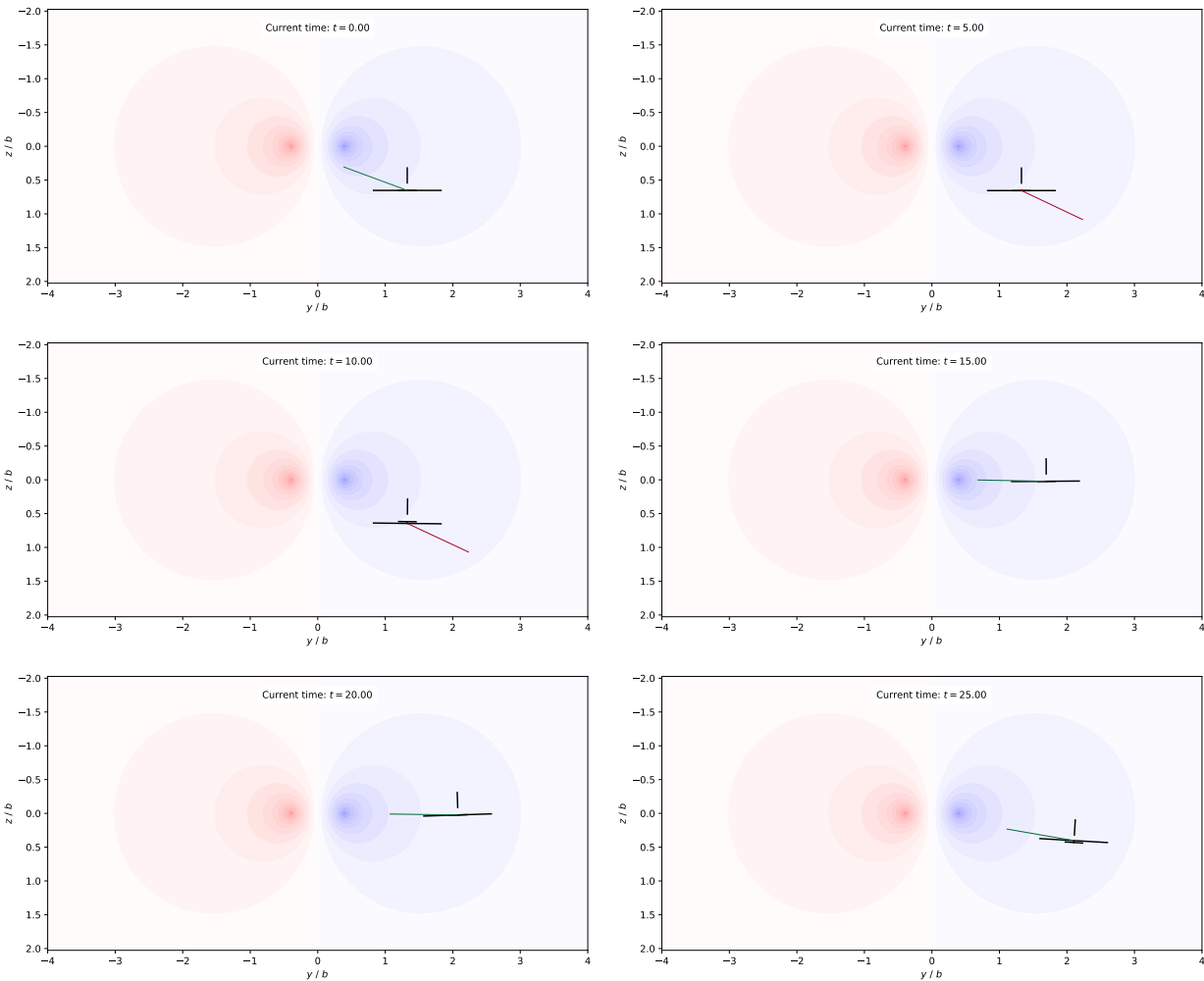
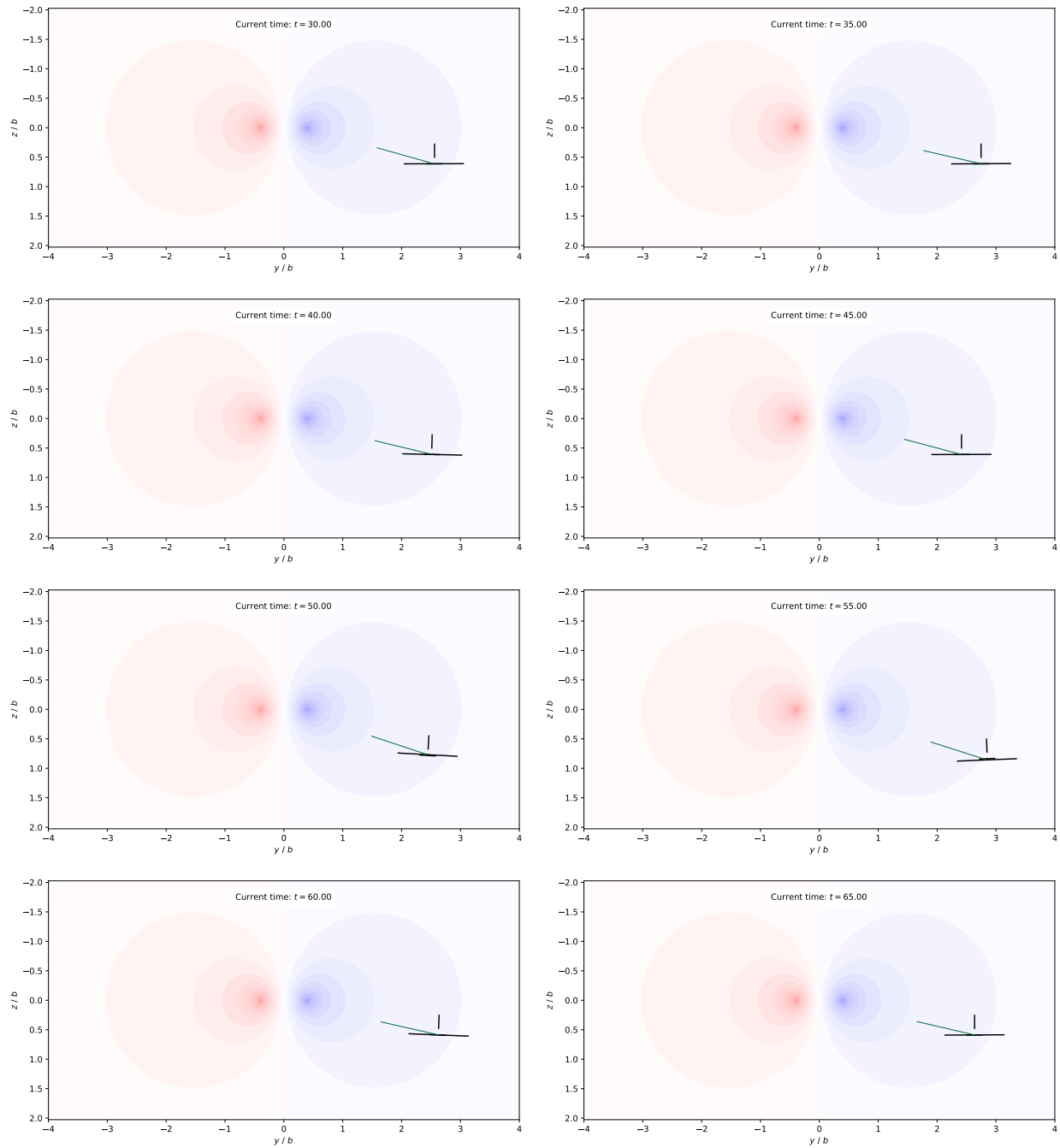


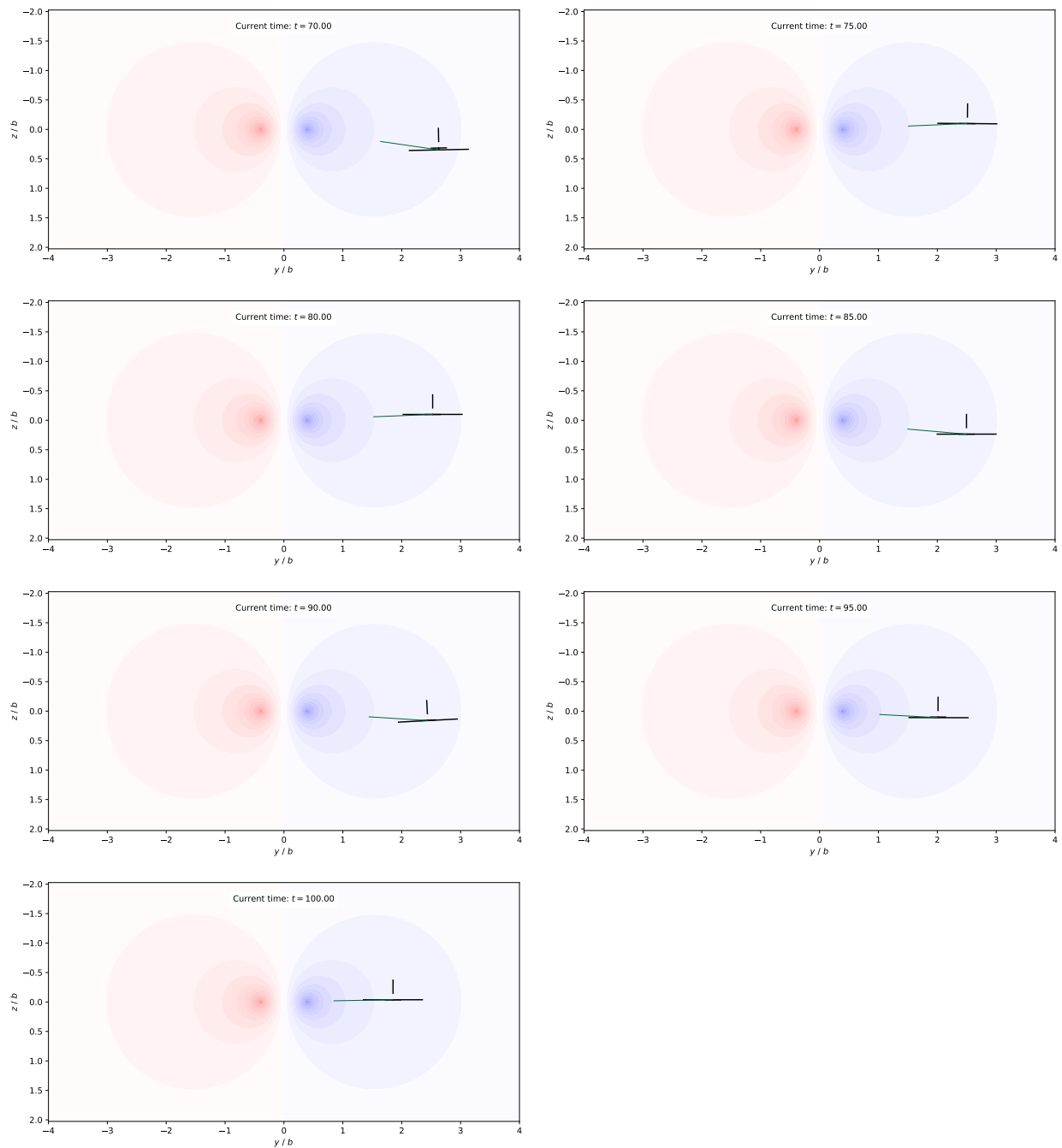
Figure C.4: Training performances of simple RNNs which aim to predict if the follower aircraft come closer to the leader, or not. Performances are represented for different amount of nodes in recurrent cells in function of the amount of previous time steps which are taken into account: plain lines (—) represent training performances while dashed lines (- - -) represent validation performances.

APPENDIX D

ANIMATION OF THE WAKE'S OBSERVATION







LIST OF FIGURES

2.1	Axes Systems ^[10]	4
2.2	Airfoil Induced Lift and Drag	5
2.3	Rear View of a Flying Aircraft	5
2.4	Motion Variables Notation ^[10]	8
3.1	Organizational overview of various machine learning methods ^[20]	11
3.2	Neuron Processing	12
3.3	Feed-Forward Neural Network	15
3.4	Recurrent Neural Network	16
3.5	One unit of improved recurrent neural networks schemes.	18
3.6	Database Subsets	21
4.1	Target Policy and Available Targets from Aircraft's Position	25
4.2	Feedback of the Environment on an Aircraft	25
4.3	Orientation and distance of the Wake in function of the Relative Position of a Follower Aircraft	26
4.4	Inputs and Outputs of the Side Prediction Model.	27
4.5	Database for the Side Prediction Model	27
4.6	Inputs and Outputs of the Orientation's Approximation Model.	28
4.7	Inputs and Outputs of the Approach Predicting Model.	29
4.8	Global view of all models, including interconnections and preprocessing phases.	30
5.1	Feed-Forward NNs' Performances for the Side Prediction	32
5.2	Simple RNNs' Performances for the Side Prediction	32
5.3	LSTM RNNs' Performances for the Side Prediction	33
5.4	GRU RNNs' Performances for the Side Prediction	33
5.5	The log of the Unobservability Index and Estimation Condition ^[30]	34
5.6	Testing Accuracy for the Side Prediction	35
5.7	Distribution of the Testing Accuracy for the Side Prediction in function of the Relative Position	35
5.8	LSTM RNNs' Performances for the Approximation of the Orientation	37
5.9	GRU RNNs' Performances for the Approximation of the Orientation	37
5.10	Error Distribution on the Approximation of the Orientation	38
5.11	Accuracy of the Side Prediction based on the Approximation of the Orientation	39
5.12	Distribution of the MAE for the Approximation of the Orientation in function of the Relative Position	39
5.13	LSTM RNNs' Performances for the Approach Prediction	40
5.14	GRU RNNs' Performances for the Approach Prediction	40
5.15	Testing Accuracy of the Approach Prediction	42
5.16	Accuracy of the Approach Prediction in function of the Error on the Approximation of the Orientation	42

5.17 Distribution of the Testing Accuracy for the Approach Prediction in function of the Relative Position	43
C.1 Feed-Forward NNs' Performances for the Approximation of the Orientation	55
C.2 Simple RNNs' Performances for the Approach Prediction	55
C.3 Feed-Forward NNs' Performances for the Approximation of the Orientation	56
C.4 Simple RNNs' Performances for the Approach Prediction	56

- [1] ICAO. Éléments actuels relatifs à la création d'un système de sécurité en matière de turbulence de sillage. Technical report, ICAO, 09 2010.
- [2] Kelly McLaughlin. Revealed: How private jet caught in the wake of a superjumbo was flipped on its back, rolled and plunged 10,000ft before pilot pulled off miracle recovery, 03 2017. URL <https://www.dailymail.co.uk/news/article-4337974/Jet-caught-wake-superjumbo-nearly-crashes.html>.
- [3] Steven Portugal. Lissaman, shollenberger and formation flight in birds. *The Journal of Experimental Biology*, 219:2778–2780, 09 2016. doi: 10.1242/jeb.148114.
- [4] D. Hummel. Die leistungersparnis beim verbandsflug. *Journal of Ornithology*, 114:259–282, 1973.
- [5] M. Beukenberg and D. Hummel. Aerodynamic, performance and control of airplanes in formation flight. In *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, pages AIAA 1998–4343, 08 1990.
- [6] W. Blake and Dieter Multhopp. Design, performance and modeling considerations for close formation flight. In *AIAA Atmospheric Flight Mechanics Conference and Exhibit, AIAA 1998–4343*, 1998.
- [7] Paul M. Doane. Autonomous station keeping system for formation flight. (10/650,606), 08 2003.
- [8] Guy Norris. Boeing/fedex 777 test confirms wake fuel-burn benefit, 12 2019. URL <https://aviationweek.com/aerospace/boeingfedex-777-test-confirms-wake-fuel-burn-benefit>.
- [9] Airbus. How a fello'fly flight will actually work, 09 2020. URL <https://www.airbus.com/newsroom/stories/how-a-fello-fly-flight-will-actually-work.html>.
- [10] Michael V. Cook. *Flight Dynamics Principles : a linear systems approach to aircraft stability and control*. Elsevier, 3rd edition, 2013.
- [11] Jr. John D. Anderson. *Fundamentals of Aerodynamics*. Mc Graw Hill, 5th edition, 2016.
- [12] A. C. Kermode. *Mechanics Of Flight*. Pearson Education, 11th edition, 2007. ISBN 9788131716571.
- [13] I.H. Abbott, I.H.A.A.E. Von Doenhoff, E.A. Abbott, and A.E. Von Doenhoff. *Theory of Wing Sections, Including a Summary of Airfoil Data*. Dover Books on Aeronautical Engineering Series. Dover Publications, 1959. ISBN 9780486605869.
- [14] L.J. Clancy. *Aerodynamics*. A Halsted Press book. Wiley, 1975. ISBN 9780470158371.
- [15] Max M. Munk. General theory of thin wing section. Technical Report 142, 1992.
- [16] Ludwig Prandtl. Tragflügeltheorie. *Mathematisch-Physikalische Klasse*, pages 451–477, 1918.
- [17] M. J. Lighthill. *An Informal Introduction to Theoretical Fluid Mechanics*. Oxford University Press, 1986. ISBN 0198536305.

- [18] Denis-Gabriel Caprace, Philippe Chatelain, and Gregoire Winckelmans. Lifting line with various mollifications: Theory and application to an elliptical wing. *AIAA Journal*, 57:1–12, 12 2018.
- [19] Ignace Ransquin, Denis-Gabriel Caprace, Matthieu Duponcheel, and Philippe Chatelain. Wake vortex detection and tracking for aircraft formation wake vortex detection and tracking for aircraft formation flight. in preparation, 2021.
- [20] Steven Brunton, Bernd Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52, 01 2020.
- [21] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 03 1989.
- [22] Yashu Seth. Which activation function to use in neural networks ?, 02 2018. URL <https://yashuseth.blog/2018/02/11/which-activation-function-to-use-in-neural-networks/>.
- [23] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.
- [24] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.
- [25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.
- [26] Ajay Ohri. Dropout in (deep) machine learning: A simple overview, 03 2021. URL <https://www.jigsawacademy.com/blogs/ai-ml/dropout>.
- [27] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*. Springer, 1st edition, 2013.
- [28] Sofiene Alouini and Sofia Calcagno. Les réseaux de neurones récurrents : des rnn simples aux lstm, 10 2019. URL <https://blog.octo.com/les-reseaux-de-neurones-recurrents-des-rnn-simples-aux-lstm/>.
- [29] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9:1735–1780, 11 1997.
- [30] Levi DeVries and Derek A. Paley. Wake estimation and optimal control for autonomous aircraft in formation flight. In *AIAA, Guidance, Navigation, and Control (GNC) Conference*, 08 2013. doi: 10.2514/6.2013-4705.
- [31] M. S. Hemati, J. D. Eldredge, and J. L. Speyer. Wake sensing for aircraft formation flight. 08 2012.
- [32] P.G. Guest and P.G. Guest. *Numerical Methods of Curve Fitting*. Cambridge University Press, 2012. ISBN 9781107646957. URL <https://books.google.be/books?id=UjnBOFIWv>.
- [33] Grégoire Winckelmans and Philippe Chatelain. Lmeca2323 – aerodynamic of external flows, 2020.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | www.uclouvain.be/epl