TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Page

clicours.com

LIST OF ABREVIATIONS

BLDC	Brushless Direct Current
CAN	Controller Area Network
CANopen	Open Controller Area Network
COB-ID	CAN object identifier
COM	Serial Communication Port
CRIAQ	Consortium for Research and Innovation in Aerospace in Quebec
DAQ	Data Acquisition
DDI	Digital Dial Indicator
DFT	Discrete Fourier Transform
EDS	Electronic Data Sheet
EPOS	Easy to use Positioning System
ÉTS	École de Technologie Supérieur
FFT	Fast Fourier
IAR	Institute for Aerospace Research
ID	Identification Number
LARCASE	Research Laboratory in Active Controls, Avionics and Aeroservoelasticity
LVDT	Linear Variable Differential Transducer
NI	National Instruments
NMT	Network Management
NRC	National Research Council Canada
NSERC	National Sciences and Engineering Research Council of Canada

OSI	Open System Interconnection
PDO	Process Data Object
PID	Proportional Integrator Derivative
PLC	Programmable Logic Controller
RPDO	Receive Process Data Object
SDO	Service data object
SMA	Shape Memory Alloy
STD	Standard Deviation
TCP/IP	Transmission Control Protocol/ Internet Protocol
UAVs	Unmanned Aerial Vehicles
UDP	User Datagram Protocol
USB	Universal Serial Bus
WWTs	Wind Tunnel Tests

INTRODUCTION

Passenger air traffic is continuously increasing around the world; some predictions indicate that 3.6 billion passengers will fly in 2016 (Green Aviation, 2010). Such a level of air traffic could be harmful to the environment given the number of flights required to fulfill the market needs. Preserving the environment is a major challenge for the aviation industry, for all of those involved in the aeronautical world. A number of research programs have been established to reduce, as much as possible, the impact of the aircraft industry on the environment (Green Aviation (2010), Popov et al. (2009) and Sugar Gabor et al. (2015)). A major focus of these research programs is the airframe technology, as it affects aircraft structure weight, aerodynamic performance, fuel consumption and noise.

Within this context, the multidisciplinary international project CRIAQ MDO 505 began in 2012 between Canadian and Italian teams. The academic team members were from École de technologie supérieure (ETS), École Polytechnique de Montréal, Canada's National Research Council (NRC), Bombardier Aerospace and Thales Canada. Two other Italian teams from the university of Naples Federico II and CIRA are also present in this project. The leader of this project is my research supervisor Dr Ruxandra Botez, director of the LARCASE Laboratory, at ETS.

The goal of the CRIAQ MDO 505 project is to reduce the drag effect on aircraft, thus the fuel consumption, through an innovative flow control concept. The upper skin of the wing will be morphed depending on the flight conditions, thereby delaying the laminar to turbulent flow transition. The control of the transition is expected to have significant economic and environmental benefits (improved aircraft energy efficiency, lower CO_2 emissions and reduced noise).

Within this project, a wing prototype with a carbon fiber reinforced flexible upper skin is designed and fabricated. This wing has an integrated morphing mechanism, an actuated aileron and pressure sensors (32 Kulite sensors) distributed chordwise. Controlling the wing morphing system, including pressure data acquisition and aileron deflection, are among the responsibilities of our research laboratory, the Research Laboratory in Active Controls, Avionics and Aero-Servo-Elasticity, LARCASE. A complete control system was integrated, tested and used during wind tunnel tests performed in the IAR-NRC's subsonic wind tunnel in Ottawa.

The focus of this thesis is to present our realizations within this project. It consists of the hardware/software integration and the control of the wing upper skin shape.

We start by reviewing the general morphing principle, morphing mechanisms and main control methods. We then present fieldbus communication, focusing on two specific protocols (CANopen, Modbus TCP) that are used for the communication with our wing actuators. Both protocol principles and their implementation details are explained.

In the second chapter, we offer a detailed description of the wing structure, including the inbox morphing mechanism as well as its control system components. A main part of our work was the implementation of communication protocols; the communication with the morphing mechanism drives is presented and demonstrated, followed by a description of another communication protocol used for the aileron.

We then dedicate an entire chapter to detail the method developed for controlling the skin displacement within a very high precision. Because of the need to address several abnormal issues during our first attempt to optimize the skin shape, we present the methods adopted to solve each issue, from the closed loop for actuator motion control to the automated calibration procedure. The last chapter covers the wind tunnel tests performed at the IAR-NRC in Ottawa, and then presents the results obtained during those tests (The controller results as well as aerodynamic improvements).

CHAPTER 1

LITERATURE REVIEW

1.1 Morphing wing

1.1.1 Morphing concept

The simplicity, the elegance and the efficiency of flight in nature have always inspired researchers and aircraft designers, especially in terms of the link between the structure and function characterizing the wings of birds (Bowman, Sanders et Weisshaar, 2002) and (Barbarino *et al.*, 2011)).

The ability of birds to change their wing shapes according to each specific flight task lead to the idea of aircraft wings shape modification. Indeed, Mujahid et Lind (2006) worked on the avian morphology principle that makes possible the use of specific wing configurations for every type of flight.

As aircraft have been designed to support important loads, their structures are unable to adapt to aerodynamic conditions change. Newer aircraft designs have underscored the need for a morphing wing and other components configurations during their flights (Stanewsky, 2001).

Morphing has therefore been defined in the aerospace field as “a set of technologies that increase a vehicle’s performance by manipulating certain characteristics to better match the vehicle state to the environment and task at hand”(Weisshaar, 2006).

The ‘history of morphing’ as described by Weisshaar (2006) shows that there were some problems associated with morphing solutions, such as cost, complexity and weight, but that they could be overcome by the use of SMART materials (Chopra, 2002). Shape modification methodologies have also been developed, firstly for military applications and then for reduced scale aircrafts such as Unmanned Aerial Vehicles (UAVs).

Changing or morphing the shape of a wing requires actuators, internal mechanisms and sliding aerodynamic surfaces (Inman, 2001). The way in which shape is changed (morphed) and at which speed, the control methodology and the sensor implementation are all fundamental morphing design elements.

1.1.2 Benefits of 'morphing'

Morphing technology offers numerous advantages for both military and civil aerospace fields; it ensures improved aerodynamic performance during cruise along with reduced fuel consumption, and thus costs. For military applications, a single morphing wing can perform different missions allowing the aircraft to widen its flight envelope and be able to have multi-role capabilities that, otherwise could require different aircrafts (Barbarino *et al.*, 2011).

1.1.3 Morphing skins

An efficient flexible skin is the key for many morphing concepts; however, its design remains challenging. The skin has to respond to different requirements: it must not only be soft enough to allow the shape changing, but it must also be stiff enough to support the required aerodynamic loads, all while maintaining the needed shape/profile (Barbarino *et al.*, 2011).

Each application has its own specific loading scenario and shape changing requirements which directly influence the design of the flexible skin. Many different materials have been tested (Thill *et al.*, 2008), and different solutions have been considered (Wereley et Gandhi, 2010) in the design of morphing skins with the aim to satisfy the various requirements.

Numerous research studies have been focused on optimizing the shape of the wing airfoil that affects the transition between laminar and turbulent flows, as a turbulent flow has a major negative effect. An increase in the drag causes more fuel consumption and therefore increasing costs. Indeed, the research realized by the LARCASE team under the supervision of Dr Ruxandra Botez within the scope of several different research projects, have produced notable results.

Projects such as the optimization of the ATR42 aircraft airfoil ((Sugar Gabor, Koreanschi et Botez, 2012), (Koreanschi, Sugar Gabor et Botez, 2015)) and the Hydra Technologies S4 Unmanned Aerial System (UAS) airfoil (Sugar Gabor et al., 2015), obtained notable transition delays of up to 20% of the chord, thus, reducing the total drag by 15%. These morphing systems has led to the improvements of aerodynamic performances for high angles of attack ((Sugar Gabor, Koreanschi et Botez, 2015) and (Sugar Gabor, A. Koreanschi et R. M. Botez, 2013)) .

1.2 CRIAQ 7.1 Project

CRIAQ 7.1 is a multidisciplinary project realized within collaboration between ETS-LARCASE team, 'École polytechnique de Montréal' and the National Research Council Canada Institute for Aerospace Research (NRC-IAR). It has been funded by the Consortium for Research and Innovation in Aerospace in Quebec (CRIAQ), the National Sciences and Engineering Research Council of Canada (NSERC), Bombardier Aerospace and Thales Canada.

The aim of the project was to improve the aerodynamic performance of a morphing wing prototype by changing the upper skin shape according to the desired flight condition. The rectangular wing was equipped with a flexible skin, smart actuators and optical sensors, and could change its shape through an active controller. Next, theoretical and experimental wind tunnel studies of the system were realized in order to validate the adopted control method and

clicours.com

to ensure the accurate transition displacement toward the trailing edge for aerodynamic performances increase (Popov, 2010).

A description of some aspects of this work is given below, including the morphing mechanism, the control system architecture and the deployed methodology.

1.2.1 Morphing mechanism

The morphing mechanism considered in the CRIAQ 7.1 project was described by Popov *et al*. (2009). The authors developed a closed control system in order to ensure the connection between the flow fluctuations located over the wing surface and the actuators which were used for the morphing operation. Kulite sensors, which are conventional pressure transducers, were used for the detection of flow fluctuation signals. By utilizing a real time data processing, it was possible to determine the transition location, so that a signal could be sent to the actuators to modify the wing surface, and thereby delay the transition location.

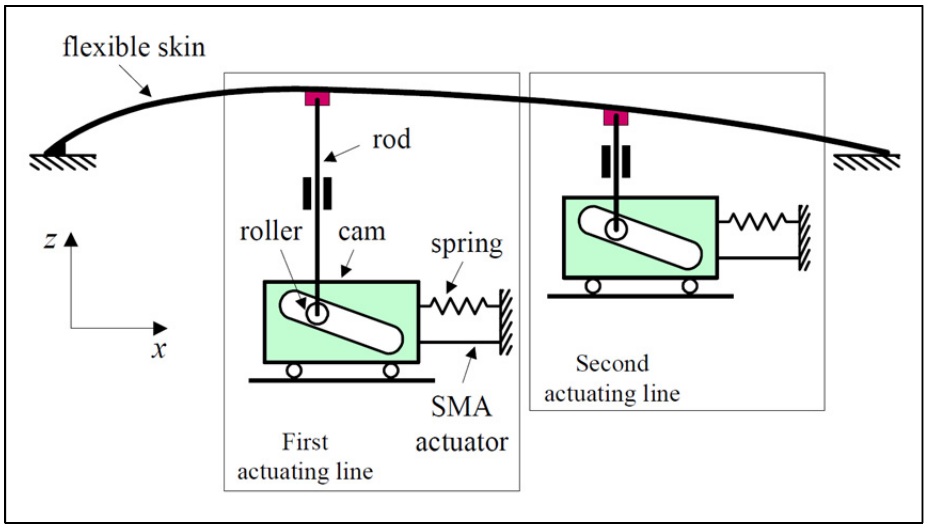


Figure 1.1 The mechanical principle of morphing
Popov *et al.* (2009)

The capability of Kulite transducers to detect small pressure variations was tested on a morphing wing in a subsonic wind tunnel at the IAR-NRC. A rectangular wing model composed of two different parts constitutes the 'morphing wing' model shown in Figure 1.1. The first part, made of aluminum, is fixed and supports the resistance forces acting during the tests. The second part is a flexible skin added on the upper surface of the wing which changes its shape according to two actuation points to realize an airfoil optimized for each test airflow condition. Two shape memory alloy actuators (SMA) were used to create the displacements of the two control actuation points. Fifteen Kulite transducers were installed on a diagonal line at 15 degrees to the center line of the wing, numbered from the location nearest to the leading edge (#1) to the one nearest to the trailing edge (#16).

Popov, Grigorie et Botez (2009) provided a detailed description of the SMA actuators. As indicated in Figure 1.1, two oblique cam sliding rods compose the actuators designed to convert the motion from horizontal along the span to perpendicular along the chord. The equilibrium between the SMA wires pulling the sliding rod in the reverse direction gives each actuator its position. When SMAs are inactive, gas springs are added to counteract the pulling effects caused by the aerodynamic forces acting in the wind tunnel. The whole model was vertically installed in the wind tunnel. Runs were then performed for different airflow conditions including the angle of attack and Mach number.

1.2.2 Control system architecture used with wing morphing

Popov, Grigorie et Botez (2009), used a specific control system architecture that was very important for implanting the bench test and for obtaining of the different experimental results in the wind tunnel. The flexible wing skin was designed and manufactured by the LAMSI team at ETS to ensure the modification of its shape accordingly to the two action points, thereby obtaining the optimized airfoils for the various airflow conditions.

Each SMA actuator had six wires powered by two AMREL SPS power supplies and controlled by a QUANSER Q8 control board via analog signals. The Q8 control board was programmed in Simulink/xPC language using an interface that allowed the user to specify the optimized airfoil shapes and the values required for the SMAs displacements. Popov *et al*. (2010) introduced details on the sensor acquisition systems developed in the SMA control architecture. Figure 1.2 shows that the acquisition of the pressure data was achieved by use of a NI-DAQ USB 6210 card with 16 analog inputs having a total sampling rate equal to 250 KS/s. The IAR-NRC analog data acquisition system was connected on one side to the input channels and on the other side to the 15 Kulite transducers. An extra channel was used, not only for dynamic pressure acquisition in the wind tunnel, but also for the pressure coefficient calculation from the measured pressure values. The signal processed in Simulink was displayed in real time.

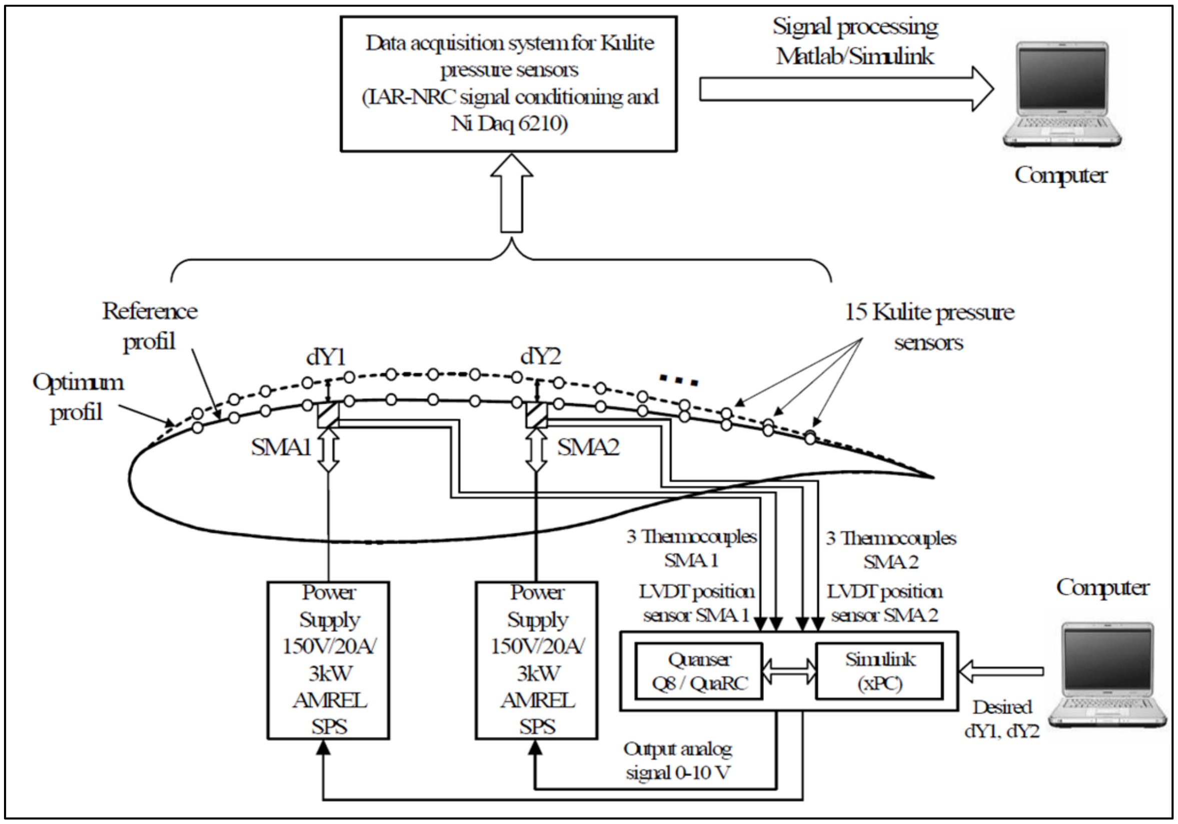

Data acquisition system for Kulite pressure sensors (IAR-NRC signal conditioning and Ni Daq 6210)
Signal processing Matlab/Simulink
Computer
Reference profil
Optimum profil
dY1
dY2
15 Kulite pressure sensors
SMA1
SMA2
3 Thermocouples SMA 1
LVDT position sensor SMA 1
3 Thermocouples SMA 2
LVDT position sensor SMA 2
Power Supply 150V/20A/ 3kW AMREL SPS
Power Supply 150V/20A/ 3kW AMREL SPS
Quanser Q8 / QuaRC
Simulink (xPC)
Computer
Desired dY1, dY2
Output analog signal 0-10 V

Figure 1.2 Sensor acquisition system in the SMA control architecture
Popov *et al*. (2010)

1.2.3 Control method used for morphing wing

In addition to the importance of the control system architecture described above, the control method used for the tests played a significant role, as it affected the entire system behavior. Popov, Grigorie et Botez (2009) proved that for each SMA actuator, there was a controller keeping it in the desired position. Figure 1.3 illustrates the controller concept composed of a PID and a switch, allowing the connection and the disconnection of the SMA actuator to a current source to heat and then to let the SMA cool, allowing it to change its length. Another method, self-tuning using fuzzy logic, has also been tested and applied to SMA control (Popov *et al.*, 2010).

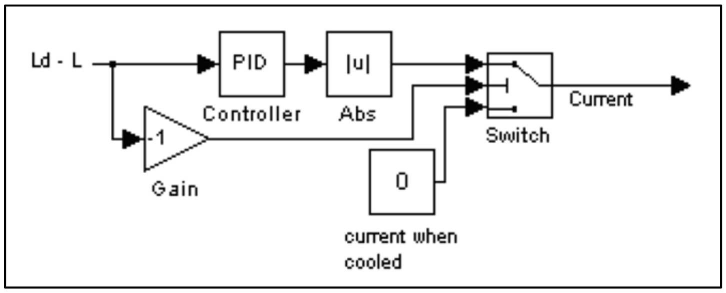


Figure 1.3 SMA current control architecture
Popov, Grigorie et Botez (2009)

1.3 CRIAQ MDO 505 Project

The studies and experiments in the CRIAQ MDO 7.1 project have led to a significant improvement of the prototype aerodynamic performance; drag reduction of as much as 30 % has been achieved (Mamou et al., 2010). In the same context, the CRIAQ MDO 505 project entitled 'Morphing Architectures and Related Technologies for Wing Efficiency Improvement' had the same goal as its predecessor project (CRIAQ MDO 7.1), but used another prototype; this prototype had with real wing tip dimensions and structure. The main concept of this project is to improve the aerodynamic performances of a wing demonstrator at low subsonic speeds by optimizing the pressure distribution, and by extending the laminar flow on the upper skin. The demonstrator is a wing-tip section which consists of a main wing box and an aileron. Design, manufacturing and aerodynamic tests were carried out over a period of three years. The wing model, equipped with a flexible composite skin, was tested with a rigid aileron, and then a morphing aileron was considered. The design and manufacturing of the morphing aileron took place in Italy, with the participation of Naples Frederico University II, Alenia and CIRA (the Italian Aerospace Research Center). The integration of the morphing aileron with the morphing wing was performed at the NRC in Ottawa during the last tests phase of the project. This thesis was realized within the CRIAQ MDO 505 project.

1.4 Fieldbus

The traditional method for connecting field-mounted devices to the inputs and outputs of a control system or a PLC (programmable logic controller) involves an overwhelming amount of wiring and connections (Blevins et Nixon, 2010). Technological improvements have led to more intelligent devices (motor drives, intelligent sensors) with embedded microcontrollers which should be able to intercommunicate and exchange data during complex control and automation operations (Ayre et Keydel, 2003). The need to replace traditional cabling and improve access to distributed information led to the introduction of Fieldbus communication (Dietrich et Sauter, 2000).

The typical definition of fieldbus is that it is ”a network for connecting field devices such as sensors, actuators, field controllers (such as PLCs, regulators, drive controllers…) and man–machine interfaces” (Thomesse, 2005). Nowadays, many different fieldbus-based communications protocols are used in industries, most of which are defined by international standards (AS-Interface, CAN (Controller Area Network), CANopen, EtherCAT, MODBUS, FOUNDATION-fieldbus, Profibus, etc).

We focus here on the description of two protocols (CANopen and MODBUS) as their implementation is required for our project.

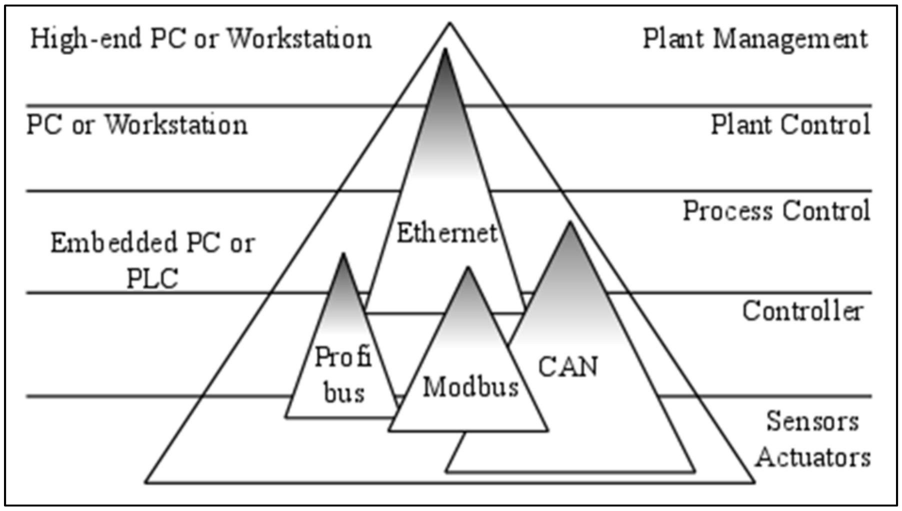


Figure 1.4 Communication protocols in an automation pyramid
Adapted from Ayre et Keydel (2003)

The Automation pyramid in Figure 1.4 shows the different control levels and the architecture of a typical automation system. The CAN, which is the basic layer of the CANopen protocol, and MODBUS protocols are present among the three lower levels among five levels; the first level contains the sensors (contact sensors, distance and pressure sensors) and actuators (electric motor, hydraulics, etc). The second level is 'the controller coupler level' which implements directly the control loops between sensors and actuators (Ayre et Keydel, 2003). The process control level contains the high-end-controllers, such as PLCs, and embedded real time controllers (PXI-e of NI as an example), which could be connected to low-level distributed controllers on the controller coupler level.

1.5 CAN and CANopen communication protocol

The standardized communication protocol CANopen is a higher-layer protocol (Figure 1.5) based on the CAN protocol, used for industrial embedded networking. Before going into the basics of CANopen, an overview of the CAN protocol is helpful in order to have a better understanding of the CANopen communication protocol.

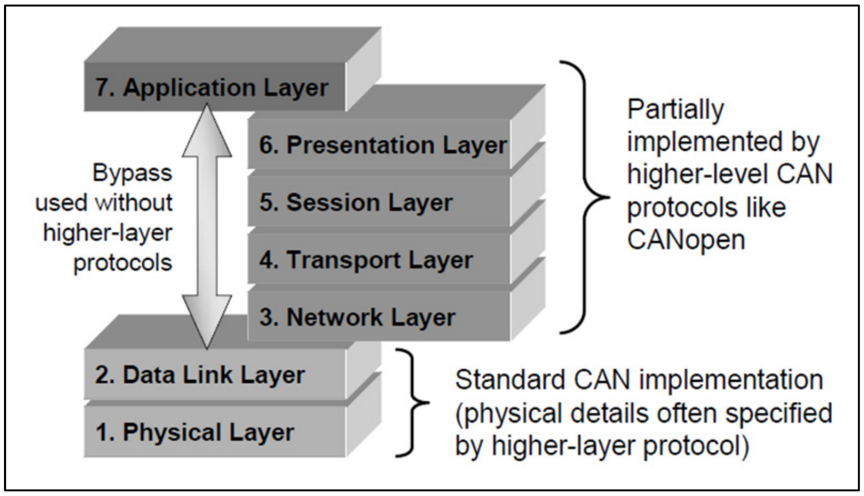

7. Application Layer
6. Presentation Layer
5. Session Layer
4. Transport Layer
3. Network Layer
2. Data Link Layer
1. Physical Layer
Bypass used without higher-layer protocols
Partially implemented by higher-level CAN protocols like CANopen
Standard CAN implementation (physical details often specified by higher-layer protocol)

Figure 1.5 Open System Interconnection (OSI)
seven-layer reference model
Taken from Ayre et Keydel (2003)

1.5.1 CAN bus

CAN is a serial communication bus protocol that uses a unique bus access control method called ‘Nondestructive Bitwise Arbitration’. “It supports distributed real-time control with a very high level of security” as mentioned by Sunit K . Sen (2014a). CAN was officially introduced by Bosch in 1986 and was intended to satisfy the high demand for electronic control systems in the automotive industry (Sunit K . Sen, 2014a).

CAN only implements the two lower levels of the Open System Interconnection (OSI) seven-layer model, the ‘Data link layer’ and the ‘Physical Layer ‘out of 7 layers (Figure 1.5). The ‘data link layer’ is responsible for packaging the received data (from the physical layer) in ‘frames’. The physical layer provide the connection of the data link layer (the CAN controller) to the physical medium (transmission medium), as it contains the CAN transceiver. The CAN transceiver has a double role, as it also converts the received signals (from the CAN controller) to a differential signal between the two wires of the network cable, are refereed as CAN_L and CAN_H (Ayre et Keydel, 2003).

A typical CAN bus consists of a twisted pair cabling with termination resistors (120 Ohm) on each side, and nodes that are connected through CAN_H and CAN_L signals(Ayre et Keydel, 2003). The use of differential signal bus makes the CAN bus very robust to external noise, thus any electromagnetic interference (EMI) is avoided. CAN bus speed can achieve 1 Mbit/s for a bus length less than 45 meters, however the bit rate decreases with the bus length(Ayre et Keydel, 2003).

A variety of higher layer protocols were developed based on CAN, such as CANopen and DeviceNet. The basics of the CANopen application layer are presented next.

1.5.2 CANopen communication protocol

CANopen is a standardized communication higher-layer protocol, implemented through the higher layers of the OSI seven-layer model. Many communication/networking technologies lack a dominant higher layer protocol standard; they only offer data transmission and reception functionalities without any specification regarding the data type or the number of messages streaming over the network (a sort of layer 2 (Data link layer) interface) (Ayre et Keydel, 2003).

In the CANopen protocol, the following parts of the higher layers (Figure 1.5) are implemented (Ayre et Keydel, 2003):

- network layer: destination addressing and routing via Service Data Object (SDO) channels, interaction functionality between a host and the network (configuration via SDO objects, etc.);
- transport layer: provides communication reliability between source and destination (provided by Network Management (NMT) services);
- session layer: defines functions, such as ‘write access’ to a shared database record (SDO channel management) and synchronization; and
- Presentation layer: involves data representations in a standardized way (Object dictionary, etc.).

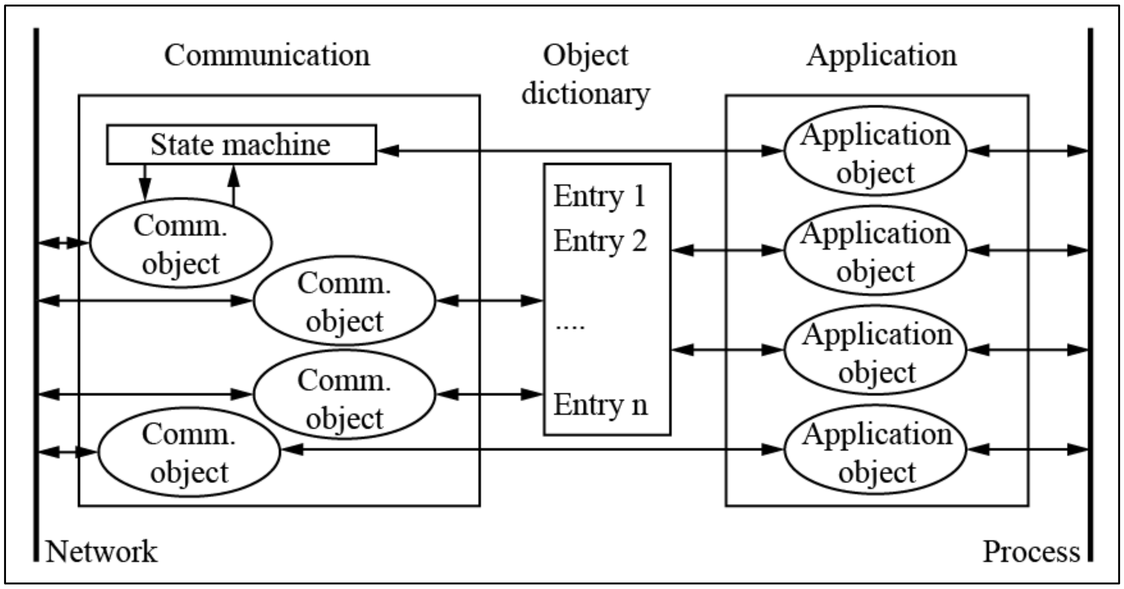


Figure 1.6 CANopen device model
Adapted from CAN in automation (2011)

A typical CANopen device model is represented in Figure 1.6. The device has three function units (CAN in automation, 2011):

- Communication function unit : provides the link between the object dictionary and the data transportation support, which is the network;
- Object dictionary; and
- Application function unit: ”the application comprises the functionality of the device with respect to the interaction with the process environment” (CAN in automation, 2011).

The terms SDO, NMT, and Object Dictionary are related to the CANopen protocol and will be defined in the following subsections.

1.5.2.1 Object Dictionary

The object dictionary is the core of any CANopen device. It is a table of objects with a 16-bit index and an 8-bit sub-index. Node information and configuration is stored in the object dictionary. All the processes and data used by the application will also be stored as entries with predefined addresses. Within the object dictionary it is possible to define the data types for each assigned variable, and to configure a node and modify its parameters by writing to the object dictionary related to that node (CAN in automation, 2011).

1.5.2.2 Service Data Object (SDO)

Since each node can implement its own object dictionary (OD), it can also implement a server for reading and writing access to that object dictionary. SDO is a direct method to access an OD using the 'Client and server' communication type (Ayre et Keydel, 2003). This method is especially useful during the initialization phase of a CANopen-based communication session, as it allows the network master to initialize and configure other nodes. There are two types of SDO objects, 'SDO read' which implement a read access and 'SDO write' which implements a write access. While it is possible to build an entire communication system by only using SDOs, for real-world real time implementation, the Process Data Object (PDO) method is preferable.

1.5.2.3 Process Data Object (PDO)

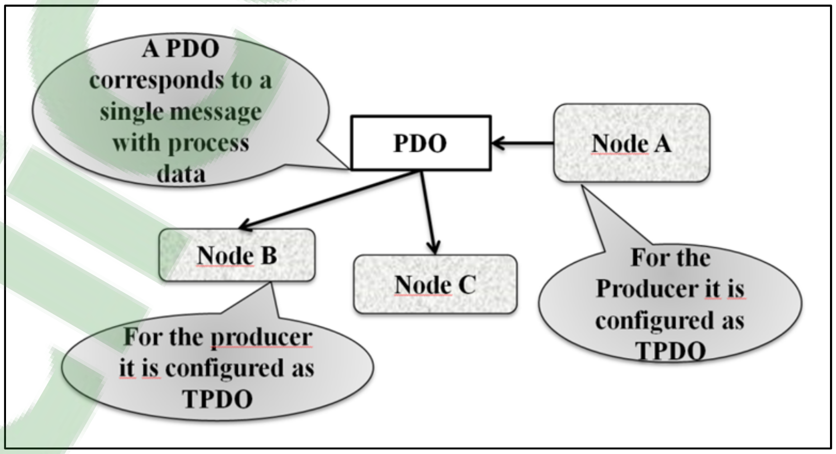

A PDO corresponds to a single message with process data
PDO
Node A
Node B
Node C
For the Producer it is configured as TPDO
For the producer it is configured as TPDO

Figure 1.7 TPDO and RPDO
Adapted from Ayre et Keydel (2003)

Clicours.com

Process Data Object provides a fast way to send real time data with a high priority. The maximum data that can be sent in a PDO structure is 8 data bytes. Each PDO is transmitted by only one node and has a unique identifier. We distinguish two types of PDO, the Transmit Process Data Object (TPDO) and the Receive Process Data Object (RPDO). When a node produces a PDO, it is considered as a TPDO for this node. For a node that receives or consumes this process data object, the PDO is an RPDO.

1.6 MODBUS communication protocol

1.6.1 Overview

MODBUS is a standardized serial communication protocol widely used in the automation industry for equipment supervision and control. This messaging structure was created by AEG-Modicon (an American company specialized in automation that is now part of Schneider Electric) for use on industrial Programmable Logic Controllers (PLCs) in 1979. It is used in multiple master-slave /client-server applications, enabling the communication between intelligent field devices and embedded controllers or PLCs (Modbus Organization, 2015).

Today, MODBUS is an open protocol which implements the application layer messaging protocol of the Open System Interconnection (OSI) model (Sunit K . Sen, 2014b). From a practical point of view, MODBUS does not require any kind of special interfaces as the CANopen protocol for example, and can be used over the Ethernet as well as with serial cables.

1.6.2 MODBUS protocol variants

There are typically three variants of MODBUS protocols:

- ASCII (the original version);

- RTU; and
- TCP/IP .

MODBUS ASCII and MODBUS RTU protocols are used with serial communication, but they are two distinct transmission modes (Sunit K . Sen, 2014b) could be consulted for further details). Three electrical interfaces may be used with MODBUS RTU devices which are the most common implementation and they are RS 232, RS 485, and RS 422.

1.6.3 MODBUS TCP/IP

MODBUS TCP is a variant of the MODBUS protocols that specifically covers the use of MODBUS messaging in an 'Intranet' or 'Internet' environment using the TCP/IP protocols (Swales, 1999).

MODBUS TCP/IP offers several advantages as seen in Sunit K . Sen (2014b) and Modbus Organization Inc (2006):

- simplicity: minimum hardware is required and it is exceptionally low-cost;
- Standard Ethernet: a standard PC Ethernet card can communicate with MODBUS TCP/IP devices. It also uses standard Ethernet cables and switches to send messages; and
- openness: it has been an open protocol since 1999.

MODBUS TCP/IP devices use Internet Protocol (IP) addressing and require a subnet mask (Sunit K . Sen, 2014b) as follows:

- IP address : "000.000.000.000" (personalized by the user); and
- Subnet mask: "255.255.255.255" (default value).

1.7 Aerospace Communication protocols

Data buses are a key element of any 'Fly by wire' aircraft control system. These networks offer many benefits to the aeronautical industry, such as reduction of the amount of wiring and thus the total aircraft weight, simple system integration as well as greater reliability and improved performance. Communication protocols on aircraft should be completely safe and reliable; they ensure the transmission of critical data between sensors, actuators and aircraft control computers (Alena *et al*., 2007). Several communication protocols for avionics have been developed, among theme we can mention ARINC 429, MIL-STD 1553B, CAN, TTP and AFDX (ARINC 664). For further details regarding these protocols, please see Alena et al. (2007) and TTTch Computertechnik (2005).

1.8 Digital PID control

PID controllers (PID stands for Proportional, Integrator and Derivative control parameters) are widely used in mechatronic systems' control and automation, and they are considered to be the control strategy most commonly used today. By some estimates, more than 95 % of control loops used in industry are PID-based loops (Åström, 2002). This control method earned its popularity thanks to its simplicity and efficiency, in which a simple manual tuning can be sufficient to obtain acceptable performance (Knospe, 2006). Several PID tuning and design methods have been developed, for further details see Wescott (2000), Gorez (1997) and Wang (2005).

Following the development of digital computers (FPGAs, microcontrollers, microprocessors), digital PID control methods have been widely adapted and enhanced with the introduction of adaptation, self-tuning, and the gain-scheduling approach to deal with highly dynamic plants shown by I.D. LANDAU et Z. Gianluca (2006) and Knospe (2006).

CHAPTER 2

MORPHING WING SYSTEM: OVERVIEW AND ARCHITECTURE

This chapter presents an overview and a short description of the wing model structure, its control system, and the adapted communication between the various components and software environments used for system integration and wind tunnel tests (WWTs).

2.1 Morphing wing model description

2.1.1 Wing description

The wing demonstrator has the real dimensions as those of a real wing tip and it was designed under real-life structural constraints. Figure 2.1 is a representation of the general appearance of the technical demonstrator; the flexible skin is delimited by the front spar and the rear spar, the wing root chord is 1.5 m and the tip chord is 1.08 m. The upper skin of the wing is made of carbon fibre reinforced composite. The morphing mechanism is fitted inside the wing box underneath the flexible skin. Section A-A (Figure 2.1) shows the difference between the non-morphed and morphed wing configurations by referring to the upper skin shape. The demonstrator is originally equipped with a rigid aileron that is replaced by a morphing aileron during the last campaign test. The morphing aileron deflection will not only substitute the rigid aileron angle deflection but should prevent flow separation above the aileron itself.

The upper skin is also equipped with a double line of 16 'Kulite' sensors each, inserted along the chordwise direction at 600 and 625 mm, respectively, from the wing tip. These sensors are used to measure the pressure distribution on the wing as well as to record flow wave frequencies for further transition studies. Pressure taps were also fixed on the leading edge, the inner surface and the aileron (rigid part of the wing) in order to retrieve pressure values around the entire wing model.

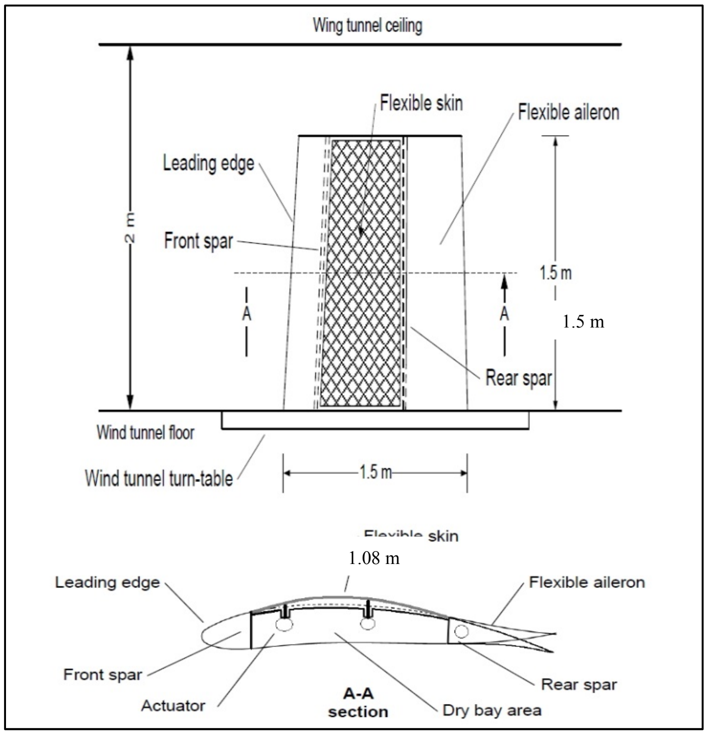


Figure 2.1 Wing structure overview

2.1.2 Model Morphing Mechanism

'Morphing' is performed via four axial actuators placed inside the wing box (Figure 2.2), and these actuators are fixed two by two to the central ribs. Each actuator is equipped with a piston, attached to the upper skin from the inside, and moves up and down. Each piston is linked to a lead screw that translates the turning motion of a gearbox into a linear motion. Brushless DC motors are used to move the gears inside the actuator mechanism and allow the piston to move linearly.

A linear variable differential transformer (LVDT) sensor is used to monitor the piston motion, and to give an accurate measurement of its displacement. An LVDT "is a type of electrical transformer used for measuring linear displacement" (WIKIPEDIA, 2015b). This position sensor is fixed in parallel with the piston (see Figure-A I-1 in the appendix) and has its tip linked to the piston so that their motion is synchronized. It is therefore possible to change the skin shape using four displacement values applied to their corresponding actuators. Next, we present the system architecture and hardware used for wing parts monitoring.

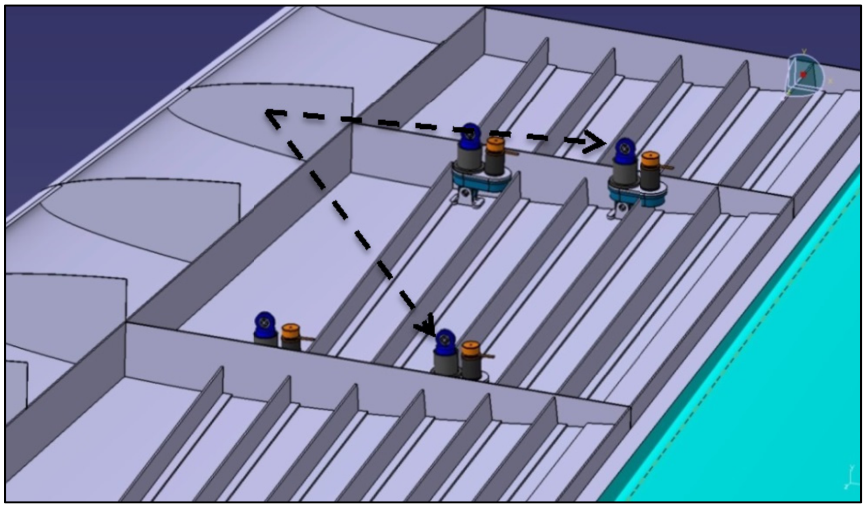


Figure 2.2 Overview of the morphing mechanism without the upper skin

2.2 Wing control system setup

The upper skin's ability to be deflected will affect the aerodynamic response of the prototype; however, operating the demonstrator during WTTs and retrieving the required experimental information will only be possible by having a very sophisticated and high-performance control and monitoring system in place. This system is the main key for the success of the experiment as well as of the entire study.

2.2.1 Real time controller NI PXI-e 8135

The core of the wing control system is the embedded real time controller PXI-e 8135 of National Instruments®. The controller runs a real time operating system, and is connected to system hardware peripherals through several input and output modules (Figure 2.3). This controller is monitored by the Host PC via an Ethernet network using the TCP/IP communication protocol, as it has a static IP address that can be personalized and fixed by the system operator. The windows machine (the Host) will serve for the control program deployment, system state control, and data monitoring in real time.

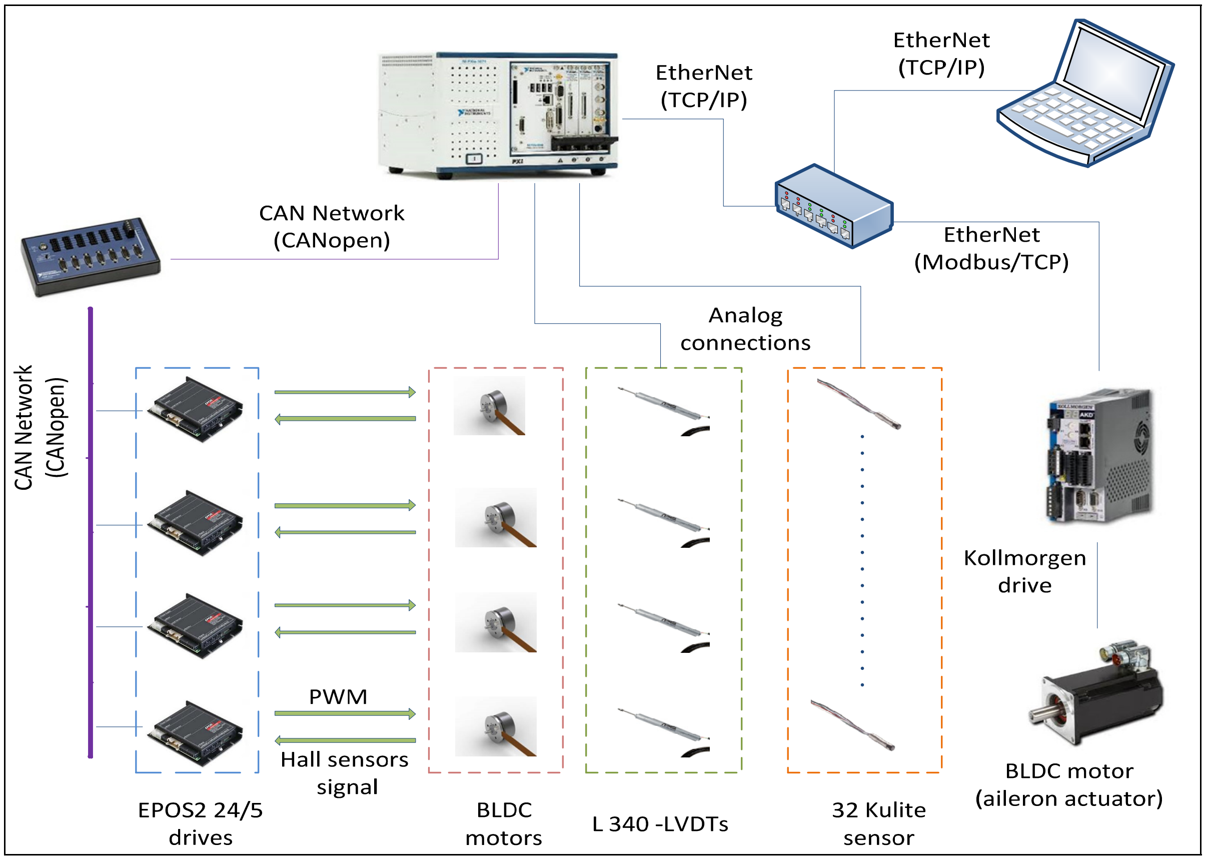


Figure 2.3 Overview of the wing control system and adopted communications between its components

However, the communication with the target is configured in such a way that losing it will not pose any danger or serious consequences regarding the running of the experiment or its actual state. In fact, all communication tasks, control and data logging will be entirely operated by the PXI, which will run independently of the Host, and its functionalities will be maintained even with no communication with the Host.

2.2.2 Fieldbus communications

Two types of industrial communication networks were utilized:

- ‘Ethernet’ is used for connecting the real time controller (PXI-e 8135), the Host and the Kollmorgen® Drive (used with the aileron actuator motor). The establishment of the communication between the host (PXI-e) and the Target (operator post) requires a TCP/IP protocol over the Ethernet, while a Modbus/TCP protocol over an Ethernet network is used for the communication between the controller and the Koolmorgen® drive. These three devices are connected via an Ethernet hub, with a distinct IP address for each.
- CAN network provides the connection between the EPOS2 24/5 drives and the controller. However, a 1-Port CANopen interface (NI PXI-8531 module) and a CAN breakout box are needed to set up the network between the Master (Controller) and the Slaves (Drives/CANopen nodes). The CANopen module is able to transmit/receive PDOs and SDOs with a speed of up to 1 Mbit/s.

2.2.3 EPOS2 24/5 drives

The EPOS2 24/5 (Easy to use Positioning System/second generation, type 24/5, 24V-5A) is a modular digital unit used for the positioning of the brushless motors with hall sensors or an encoder; however, it is also suitable for use with brushed DC motors with encoders (Maxon motor control, 2015). Rather than looking at it as a black box, we will need to know (but only to a limited extent) the basics and principles of the working of the drive and its internal architecture functioning.

The drive is built with several communication interfaces and an ‘I/O’ module for digital and analog signals (Figure 2.4). Motor phases and hall sensors are connected on the J2 and J3 modules, respectively. Specially designed to be commanded as a ‘slave’ in a CANopen network, this smart device has a CAN interface (J7/J8). In addition, it can interface with any USB or RS 232 connection (J9 and J6 respectively).

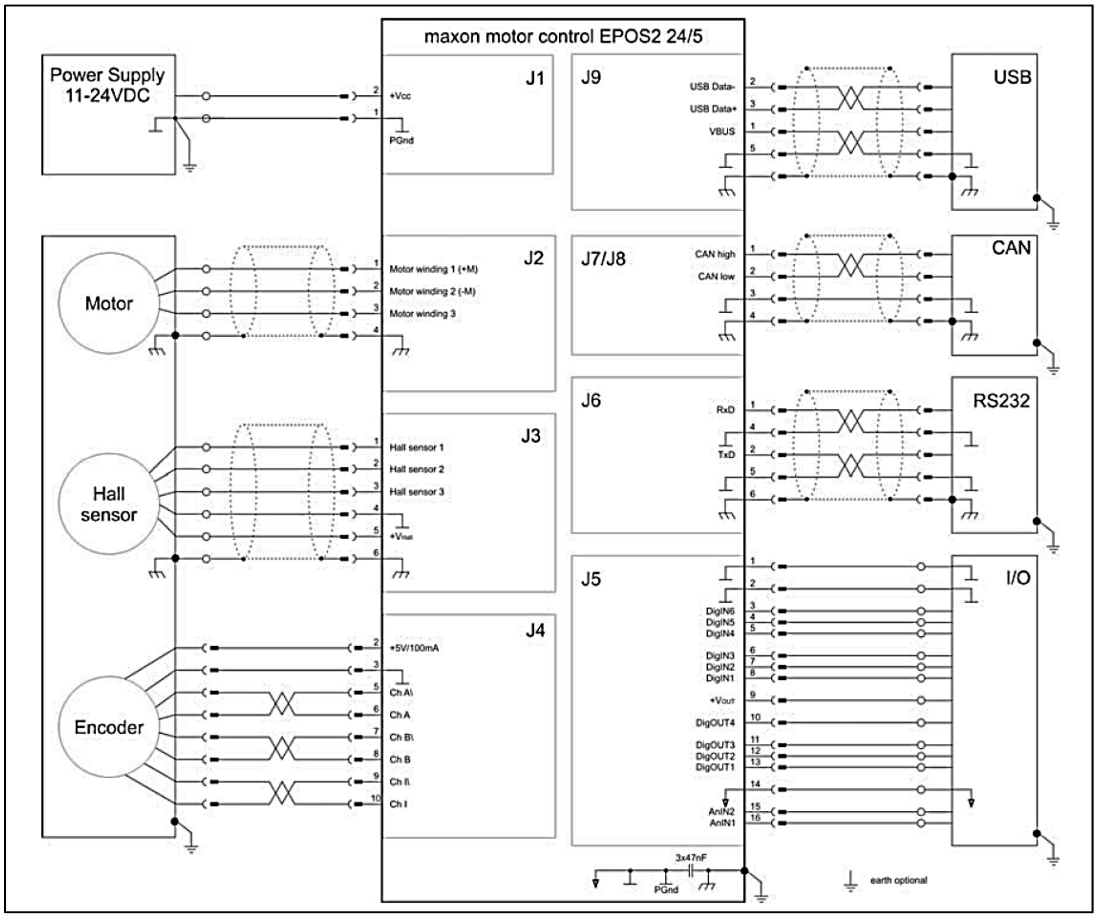


Figure 2.4 Basic internal architecture and interfaces of the EPOS2 24/5
Taken from Maxon motor control (2013b)

The EPOS2 24/5 software makes it possible to tune the embedded controller parameter via the auto-tuning functionality; moreover this is the method we use to tune the drive since it is fast and reliable. However, we cannot guarantee that use this tuning achieves the optimal regulation parameters (Maxon motor control, 2013a).

The drive ensures the electronic commutation of the brushless motor based on feedback from three hall sensors. The regulation architecture is composed of a current loop (inner loop), velocity loop and a position loop.

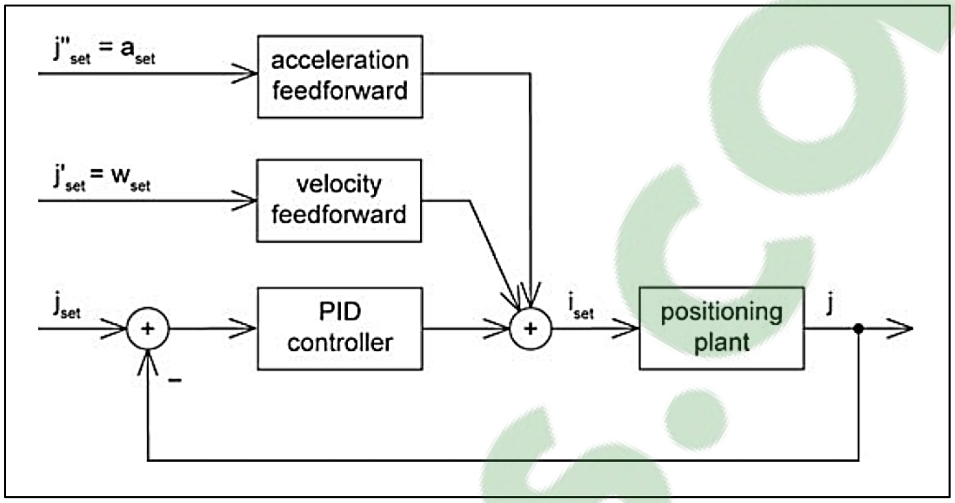


Figure 2.5 EPOS2 24/5 position regulation structure
Maxon motor control (2014b)

Figure 2.5 shows the ‘position regulation control’ loops. The inner loop, based on a PI controller, is the ‘current’ regulation loop and does not appear in this representation (positioning plant). The ‘position’ loop is based on a PID controller, with acceleration and velocity feedforward gains.

Another interesting aspect of this smart drive is the varieties of operating modes it offers. In fact, the drive’s behaviour and its functionalities can be selected in accordance with the application’s needs. It is possible to configure the drive in ten (10) different modes, including ‘profile position mode’, ‘homing mode’ and ‘position mode’ (Maxon motor control, 2014b). For our application, the ‘position’ mode is selected as it is both easy to implement and matches our desired functionalities.

2.3 System Software Setup

The main software used for the system control is NI-VeriStand®, which is widely used for real time applications and testing. A project in NI-VeriStand® (Figure 2.6) is composed of at least of three components: a ‘project file’, a ‘system definition file’ and a ‘screen file’ (National Instruments, 2015).

clicours.com

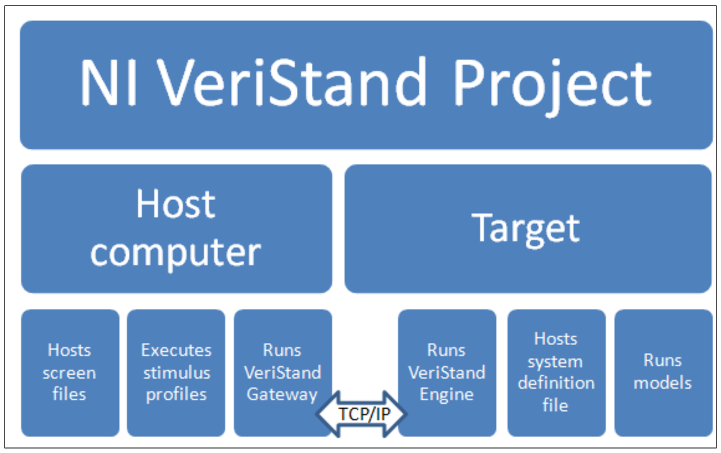


Figure 2.6 NI-VeriStand® project
Taken from National Instruments (2015)

- Project file

  Defines the high level settings such as the system definition files, the workspace (screen file), configure and run stimulus profiles, configure services, add alarm responses, etc;

- System definition file

  Contains the configuration settings of the NI-VeriStand® engine as the target rate, real time engine, hardware (DAQ, FPGA…), custom devices, models, system mappings, etc. A brief review of the main parts in the System Explorer Window of the system definition file is presented here:

  1) Target: it contains any target a user wishes to add; after adding the target, it is possible to fill in the target specifications. In the system explorer tree, the user can add the hardware list, custom devices, simulation models, calculated channels, etc…

  2) Custom devices: are a means to extend and customize the NI-VeriStand® environment. These modules are a sort of LabVIEW® code packaging that can be added to the system definition file.

  3) Simulation models: are used to communicate with other system parts that are present in the system definition; real time control operation can be implemented via the model's inputs and outputs. Once the project is deployed, the inputs and outputs will be updated each time the model is executed (the frequency of running the model is customizable).

4) Calculated channels: contain new values calculated on the base of other system channels.

5) System mapping: makes it possible to establish connections between source and destination channels. This mapping is especially useful when the output of a simulation model needs to be linked to the input of a custom device and vice versa.

- Workspace:

  Defines the configurations and settings for the screen; visualization panels and controls can be added, and all system channels and inputs can be controlled or monitored from this screen. For this project, another labVIEW® based interface was developed for the purpose of wind tunnel testing.

Table 2.1 Software environment description

Software name	Description
NI-VeriStand® 2013	Used for real time applications and testing, the main project is deployed on the real time target using NI-VeriStand®
LabVIEW® 2013	Used to develop models for NI VeriStand®. All the packaged code used for developing custom devices is the LabVIEW® code
Matlab® Simulink 2013	Used for developing control models for NI-VeriStand®

2.4 CANopen communication protocol implementation: Application to the EPOS2 24/5 drives

In this section, Details, on the implementation of the CANopen communication protocol, are presented. The CAN network and its connection with nodes is presented, and then we explain the programming side of the application with an overview of the internal architecture of the CANopen interface.

2.4.1 Physical structure of the CANopen network

The CANopen network (Figure 2.7) is composed of a total of five nodes. The real time controller is the Master node; it is connected via the CANopen module 'NI PXI-8531' (Section 2.2.2) and its supervising nodes. Four EPSO2 drives are connected in cascade (the network wiring output of one node serves as the network wiring input of the next node). Each of the four drives is considered to be 'a slave' node as it provides services under the control of the Network Master.

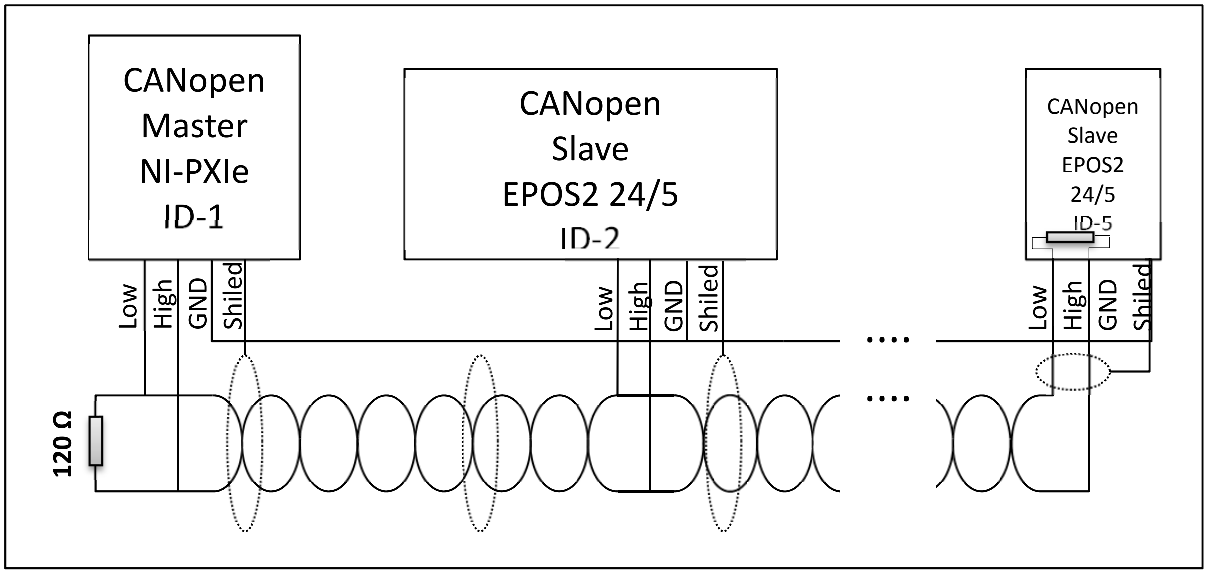


Figure 2.7 CANopen network structure

The network is terminated on each end by 120Ω resistors. On one end, the resistor is on the NI CANopen breakout box (Section 2.2.2), and the other network end is terminated by a DIP switch on the last node of the network (Figure 2.8). The DIP switch should be set to 'OFF' for intermediate nodes.

A unique 'Node Identification Number' (node ID) must be assigned to each node. Table 2.2 shows the corresponding node ID of each network device.

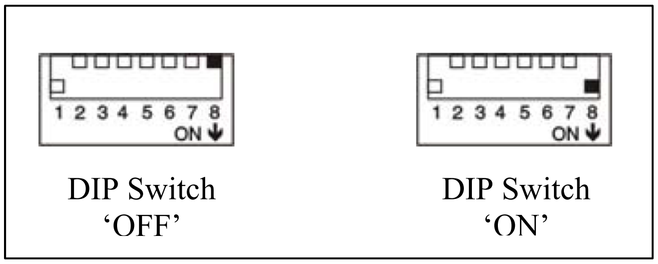
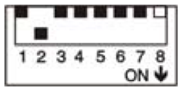


Figure 2.8 DIP Switch-CAN Bus termination
Adapted from Maxon motor control (2014a)

Table 2.2 Nodes identifier mapping

Node	Node ID
Real time controller (Master)	1
EPSO2 24/5 drive of actuator '1' (slave)	2
EPSO2 24/5 drive of actuator '2' (slave)	3
EPSO2 24/5 drive of actuator '3' (slave)	4
EPSO2 24/5 drive of actuator '4' (slave)	5

Table 2.3 Node-ID calculation
Adapted from Maxon motor control (2014a)

Node ID	DIP switch n°	1	2	3	4	5	6	7	Calculation
2		0	1	0	0	0	0	0	$2^1+0=2$

The CAN ID may be set by using the software or with the hard DIP switches present on the drive (1...7, see Figure 2.8). An address from 1 to 127 can be assigned by binary code. The CAN ID is determined by the sum of the values of DIP switches set to the 'ON' position. An ID value equal to '2' on a DIP switches can be obtained by setting the DIP switch number 2 to 'ON' and keeping the others in the 'OFF' position (Maxon motor control, 2014a).

2.4.2 Position mode

The position mode is simple to implement as it needs a minimal amount of configuration and only few parameters. It only requires a new absolute position value, which can be reached with the maximum acceleration and speed values (Maxon motor control, 2014b).

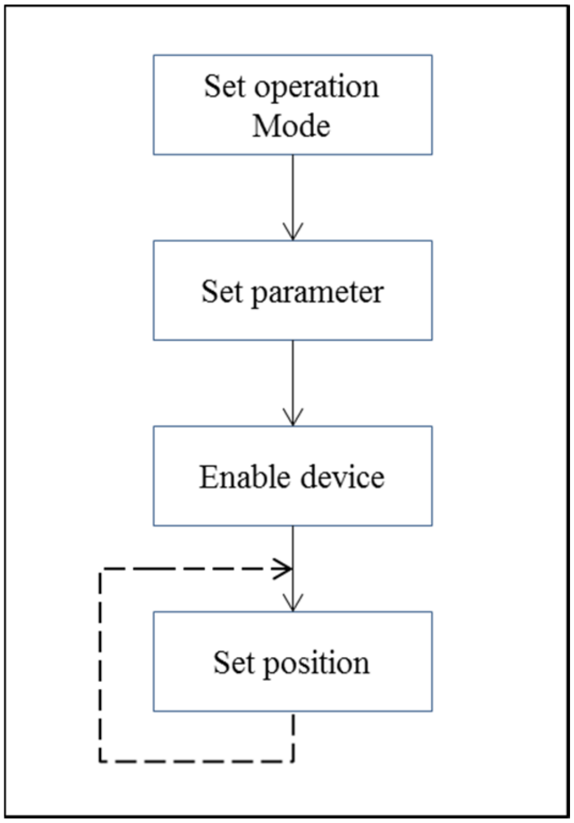


Figure 2.9 Position Mode Flowchart

Figure 2.9 shows the command sequence for configuring the drive to the ‘Position mode’. The first step consists in setting the operation mode; the second step consists on setting parameters such as the minimum and maximum position limits, maximum velocity and acceleration values. These values make it possible to apply a setting value after the drive is enabled. These commands are based on writing/reading to the drive's object dictionary through the CANopen interface. Further details are given in the next subsection.

2.4.3 CANopen implementation

Implementation of the CANopen communication is performed via a premade custom device. This custom device was developed for a specific client's application by engineers at National instruments, it may be provided by NI® upon request.

The custom device reduced our task complexity considerably, as it manages several crucial aspects of the communication. Moreover, it offers an easy to use interface with clearly-defined sections and functionalities.

2.4.3.1 Electronic Data Sheet file

The Electronic Data Sheet (EDS) plays a key role in configuring a CANopen device, as it is a standardized electronic description of the object dictionary. It contains the structure of the object dictionary, and all objects used by the manufacturer of the device. Most importantly, it allows different CANopen software producers access to the device in order to see the object dictionary or to configure it (Schmidt, 2012).

The EDS file can be generated via the EPOS software in EPOS2 drives. It can then be used with the custom device as needed for configuring the EPOS2 drives via NI software. The EDS file path should be specified on the main page of the custom device. The baud rate of the communication is established to 1 Mbit/s. Then, one could start adding device nodes with their IDs. For each node, it is possible then to configure or to add Network Management, Service Data Objects (1.5.2.2) and Process Data Objects (1.5.2.3).

2.4.3.2 CANopen Network Management (NMT)

CANopen network management operates with a Network Master (NMT master) that controls the node's states on the network (NMT slave), as it follows a Master/Slave structure.

Network management provides several functionalities, including error control services for the initialization of NMT slaves, error control services for node supervision, and communication status and nodes configuration services (Maxon motor control, 2013b).

A node can be switched into three different states:

- Pre-Operational;
- Operational; and
- Stopped.

When a device is powered and initialized, it automatically implements a state machine that brings it to the ‘Pre-operational state’. This fact is very important, because it is only in this state that a node can be configured and parameterized via Service Data Object (SDO) sequences. Process Data Objects (PDOs) are not allowed.

Using the NMT protocol, a master can change the machine state of any slave device from the ‘pre-operational’ to the ‘operational’ state and vice versa. Using the custom device, it is possible to implement this feature by choosing the mode in ‘in run state’ before configuration. In the ‘run state’, the nodes (EPOS2 drives in our case) should be in the operational state, so that it is possible to send in/out data via PDOs, and the initial mode before configuration should be ‘pre-operational’ in order to allow the drive to be configured.

For further details regarding the NMT service and functionalities for each state, please see Appendix II.

2.4.3.3 SDOs Initialization Sequence

Typically, this is the most important phase when implementing the communication system. In this step, the drive will be programmed and prepared to perform as desired during control operations.

Using Service Data objects (SDOs), it is possible to access the object dictionary, the core of the smart device. A peer-to-peer communication channel is created between the client and the server allowing reading and writing of the object dictionary entries.

Using the SDOs initialisation sequence is advantageous in our system, as it allows us to adopt the functionalities and drive behaviour to the application's needs. SDO initialisation makes it possible to choose the working mode (Section 0), to set acceleration/velocity/position limit values and to select which of the process objects are the most determinant. This last point is obviously crucial for our application, as the nature and the amount of the information exchanged on the CANopen network should be determined correctly, otherwise the real time capabilities of the CANopen communication could be significantly deteriorated.

Our application will need to be able to:

- Configure the drive into the position mode, as mentioned in Subsection 0;
- Control the drive status (enable/disable) or to reset the drive in error cases, and to set the position value for the motor;
- Retrieve the information required for the position control loop, such as, for monitoring the motor state:
    1) Drive status,
    2) Actual position value of the motor,
    3) Current actual value average, and
    4) Velocity actual value average; and
- Define limits for the acceleration, velocity and position.

Figure 2.10 shows the principle of the drive configuration and programming adopted for the initialization sequence, which is implemented by writing a specific value into the appropriate address of the object dictionary.

For example, the first task during the configuration steps is to ensure that the drive is cleared of any errors, and, is disabled (disable power to the motor and drive function). This specific operation is performed by writing a specific value to the address ‘6040h’ (addresses are given in hexadecimal in the EPOS2 documentation). For each desired state of the drive (not to be confused with the state machine), there is a value that should be written to the ‘Controlword’, and thus the drive state determines which commands are applicable to the drive (Maxon motor control, 2014c). The drive states and their corresponding control word values are given in Appendices II and III.

The second step consists in selecting the desired mode in which the drive will operate. This action is realized by writing ‘-1’on the address ‘6060h’ which corresponds to the ‘mode of operation’ parameter in the object dictionary (see appendix VI). However, in the list of SDO sequences given in the ‘Appendix VI’ the value given is indicated as ‘FF’ in hexadecimal and not in decimal format as in the drive documentation. The last four steps in the flowchart are performed in the same manner as for the operation mode selection; each parameter is accessed through the object dictionary with its address, and then the correct value is assigned and so on.

In addition to their use to access the object dictionary, SDOs are also useful to map process data channels. These channels are used to retrieve real time data from each drive, such as the actual motor position and the motor velocity values.

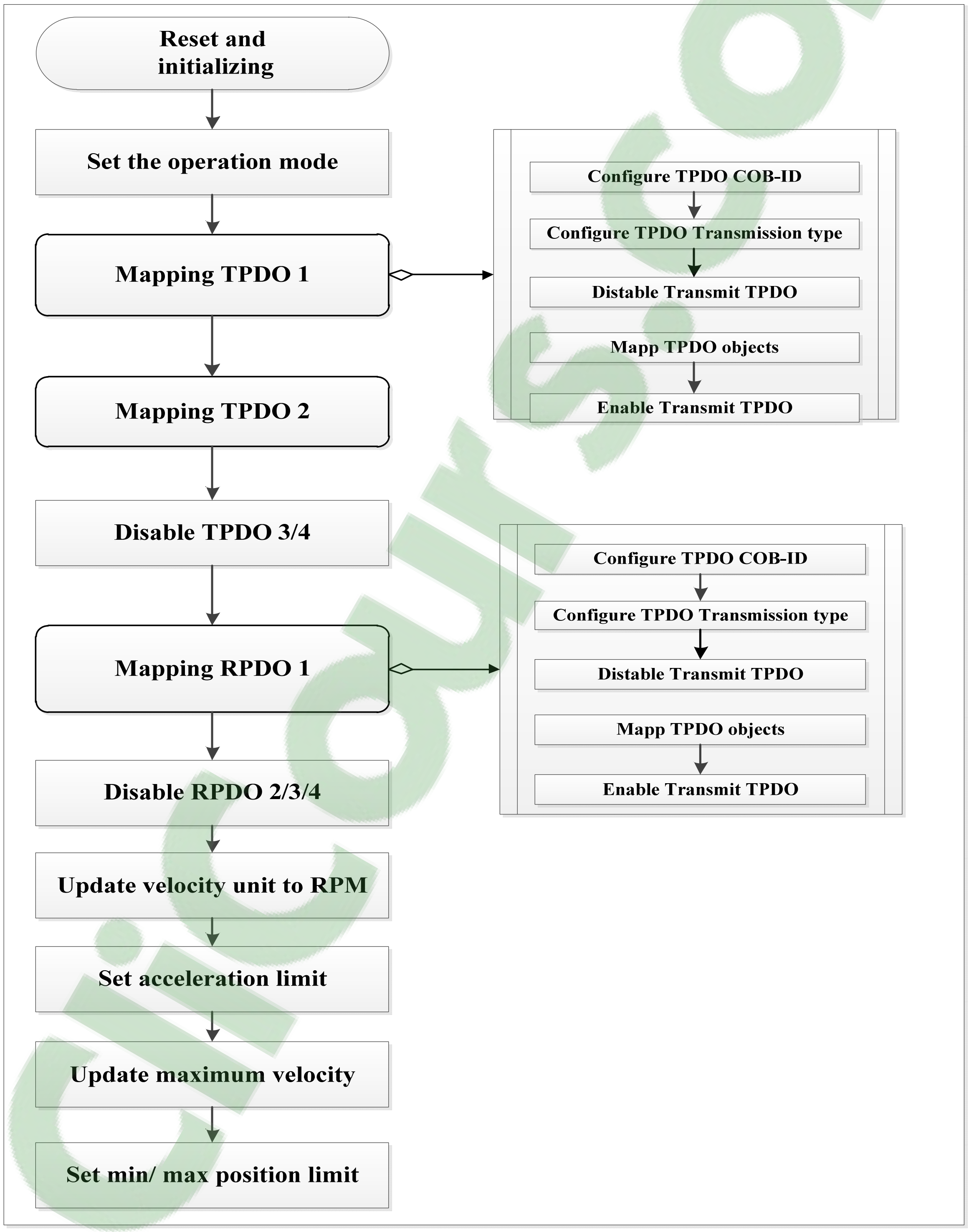


Figure 2.10 Drive configuration and initialization sequence

Clicours.com

2.4.3.4 Mapping TxPDOs and RxPDOs

A rapid way in which is broadcasted data via the network is to use ‘Process Data Objects’ (PDOs), which are high priority messages that can be transferred from one node to another node or to multiple nodes on the same CAN network. Despite being a non-confirmed messaging service (as is the case for SDOs), they are an efficient way to send real time data between nodes (Maxon motor control, 2014b).

In a PDO communication process, are distinguished two PDO services:

- Write PDO Service
- Read PDO Service

The concept can be summarized as: ”The producer sends a Transmit PDO (TxPDO) with a specific identifier that corresponds to the identifier of the Receive PDO (RxPDO) of one or more consumers” (Maxon motor control, 2013b).

As an example, let us suppose that we want to send the motor’s current value back to the CANopen master in real time. If we first map the object address which corresponds to this value’s process object into a TxPDO structure then it can be transmitted in real time.

Three types of PDO transmission modes , can be, generally, distinguished (Maxon motor control, 2014b):

- remotely requested;
- event triggered; and
- synchronous transmission.

For details about each type of transmission mode, please see the ‘EPOS2 application Notes Collection’. We are only interested in Synchronous transmission as it is the one adopted for our application.

In synchronous mode, PDOs are transmitted periodically. In fact, a ‘Sync’ message is required in a cyclic fashion in order to initiate the simultaneous sampling of the mapped parameters.

‘PDO mapping’ defines the application objects to be transmitted via a PDO. Mapping an object’s structure is described within the object dictionary of each EPOS2 device. Four TxPDO mapping objects are present from the address ‘1A00h’ to the address ‘1A03’ within the object dictionary, and it is possible to map three entries onto each TxPDO mapping object. The addresses of RxPDO mapping objects and parameters are given in Appendix V.

To map an application object (not all objects can be mapped), some steps should be followed, as described in Maxon motor control (2014b):

1) Configure the COB-ID (CAN object identifier);
2) Set the transmission type;
3) Disable the PDO transmission;
4) Map the desired object within the PDO object (multiple objects maybe mapped within one TxPDO or RxPDO); and
5) Enable PDO transmission.

These steps for mapping application objects are only briefly outlined; for further details and explanations, please refer to Section 10.5.3 entitled ‘PDO configuration’ within the ‘EPOS2 application Notes Collection’.

Table 2.4 Mapping of the ‘Statusword’ and ’Actual position value’ of node ID ‘1’

Step	Command	Data-type	Index	Sub-Index	Value (hex)
1	SDO write	U32	1800	1	182
2	SDO write	U8	1800	2	1
3	SDO write	U8	1A00	0	0
4	SDO write	U32	1A00	1	60410010
5	SDO write	U32	1A00	2	60640020
6	SDO write	U8	1A00	0	2

Table 2.4 shows an example of mapping two TxPDOS objects related to the node with ID '2', which is the 'first' motor drive. A description of each step is given below:

1) Configure Transmit PDO1 COB-ID;
2) Configure Transmit PDO1 Transmission Type to Synchronous;
3) Disable Transmit PDO1 by writing 0 to the number of mapped PDO objects;
4) Map the first object of TxPDO1 (StatusWord);
5) Map the second object of TxPDO1(actual position value); and
6) Enable transmit TxPDO1 by writing the number of mapped objects in the field with sub index 0X00.

A complete list of the SDO sequences and the mapped objects for each node are included in Appendix VII. The same functionalities were implemented on each node, and the differences between each node's programs are highlighted in blue. Basically, the differences are related to the RxPDO and TxPDO configurations, and more precisely, to the step of configuring the COB-ID value because this value depends on the node ID value (COB-ID = PDO-Offset + Node ID) (Maxon motor control, 2014b).

2.5 Modbus TCP/IP communication protocol implementation: Application to the Kollmorgen® drive

2.5.1 Overview

After implementing the CANopen communication protocol, our second challenge was to establish a robust communication system between the system controller and the aileron actuator drive. At the beginning, we thought that the communication would be CAN-based since we are already familiar with the protocol and its implementation, but it became clear that this option was not possible as we needed to overcome several technical problems.

The aileron deflection range is between [-6…+6] degrees, with a step of 1 mm. The aileron's motion is fixed and predefined within some aerodynamic limitations and security requirements. In total, 13 possible deflection angles were designed and calibrated to be

executed. An especially useful feature of this smart drive is the ‘Motion task’ for executing moves.
A ‘motion task’ is a combination of several parameters defining a move. To describe a motion task, there is a need to define the desired displacement, the velocity, accelerations and some other parameters. Via the commercial workbench, it was possible to program and load a complete motion task table to the drive. We used a 13-motion task table that was programmed and then loaded to the drive.
After the drive has been programmed, all what was needed to execute a specific aileron angle is to call up the correct motion task number. We were not able to call a motion task with the CANopen protocol, and therefore we changed our communication protocol to Modbus TCP which does not require any special interface and is easy to implement using the LabVIEW® library for Modbus communication.

Our principal requirements concerned the establishment a reliable communication with the drive, its configuration it and also to call up a motion task in order to execute a required motion. Starting from these requirements, we developed a LabVIEW® program that established communication between a windows machine and the drive. After performing this task, the developed code was packaged within a custom device and finally integrated into the project to control the drive through the system controller (NI PXI-e).

The next subsection summarizes the implementation of the LabVIEW® program and its working principles.

2.5.2 Modbus TCP implementation

Communication within the drive was established by creating a connection between the Master (PXI-e) and the Slave (Drive) using the IP address and a port number (502 for the Modbus protocol). After the establishment of the communication, it is possible to access to drive registers for reading or writing operations. However, only two Modbus functions are supported by the drive (Kollmorgen, 2014):

1) Read Holding registers and
2) Write multiple registers.

The Modbus LabVIEW® library offers VIs that can implement reading and writing operations when specifying the register address and the value to write if it is a writing operation.

Figure 2.11 shows the flow principle of the LabVIEW® program used for controlling the drive. As mentioned previously, the first step is to create a standard TCP master instance which will look for a slave at the specified IP address and port number. The drive status can then be retrieved by reading the Modbus parameter 'MODBUS.DRVSTAT'. This parameter contains important information, such as the state of the drive (active/ not active), hardware limit switches states, etc.

Next, the drive can be enabled/disabled by acting on the 'MODBUS.DRV' register, once enabled, it is possible to set the command source of the drive from among five types (Kollmorgen, 2014):

- Service, TCP/IP command (register value = 0);
- Fieldbus command (register value = 1);
- Gearing command (register value = 2);
- Analog command (register value = 3); and
- Program command (register value = 5).

For our application, the 'Service, TCP/IP command' is set as the command source mode since we are implementing Modbus communication over a TCP/IP protocol.

Next, we set the motion mode to 'Position mode', as we plan to use the motion task to execute drive motions. However, before executing any motion, it is crucial to check the reference position of the aileron. The reference position corresponds to a zero angle of deflection. This position is calculated; it is situated at the mid-point of the displacement range delimited by two hardware switches. The homing procedure is preconfigured (there are several homing strategies from which to choose), and within the Modbus communication program, executing the homing procedure can be done in one step by writing to the ‘HOME.SET’ register.

After mapping ‘HOME.SET’ register, the motion task object (MT.MOVE) is mapped when the motion task is called by number via its register. To determine whether the motion task is finished or not, we mapped the ‘DRV.MOTIONSTAT’ register, which returns the motion status of the drive. It is not possible to call a new motion task until the current motion has been completed.

A complete list of the parameters and their descriptions is given in APPENDIX VII.

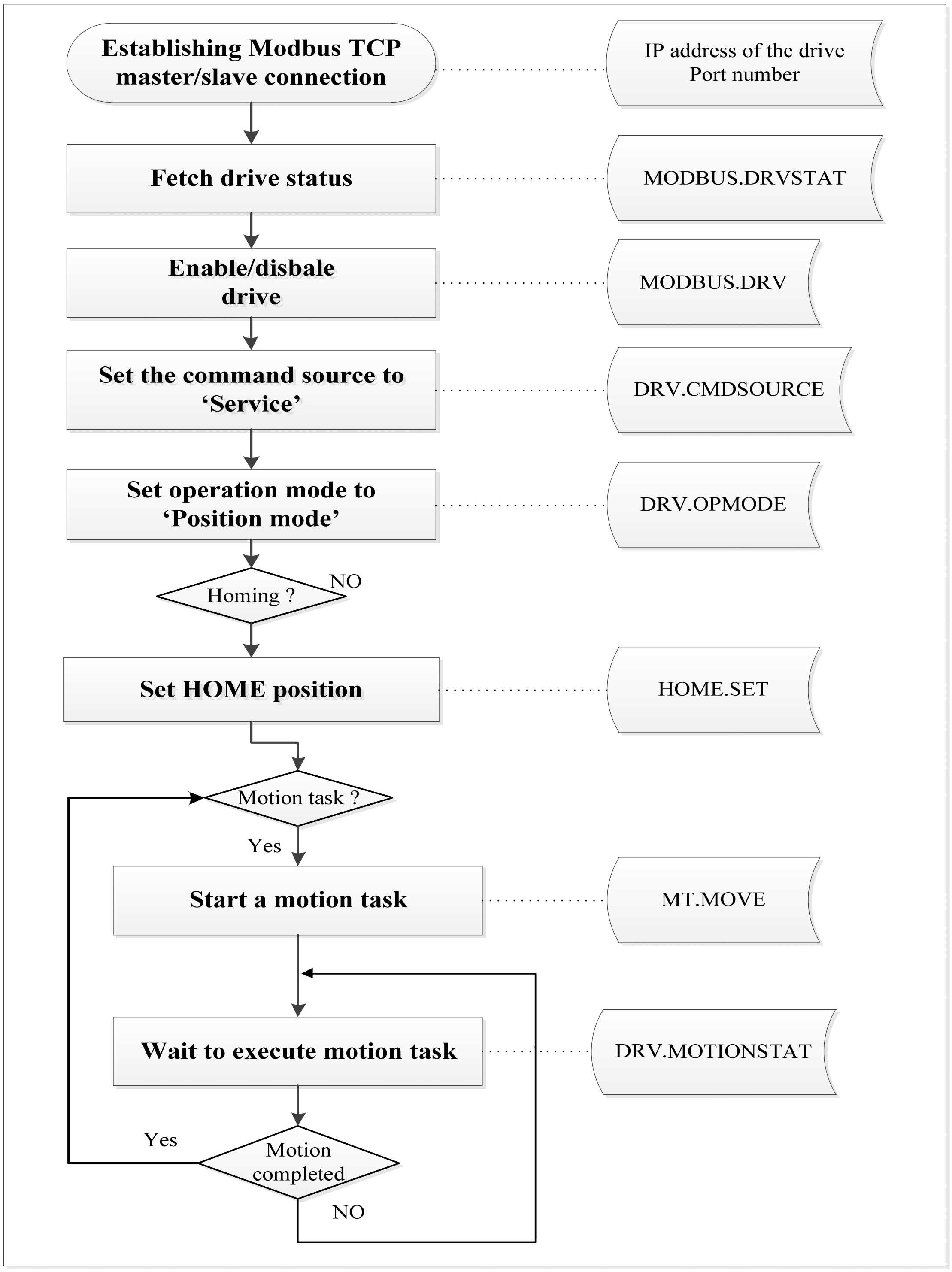


Figure 2.11 Aileron control program diagram

CHAPTER 3

WING SHAPE MORPHING

3.1 Wing morphing in open loop

3.1.1 Skin shape open loop control

As seen in the subsection 2.1.2, the mechanism used to morph the skin (APPENDIX I) is essentially composed of two parts: 1. the actuation mechanism (which contains the BLDC, gear box, lead screw and piston) and 2. the actuator displacement sensor (LVDT). When designing this mechanism it was thought that skin displacement would be linearly dependent on morphing mechanism displacement, but when we started working on the wing prototype, some unexpected problems appeared, thus making the control task more delicate and complicated.

It was believed that the desired skin displacement would be obtained by moving the actuators with the needed exact value (displacement of the piston in mm = number of motor rotations/100); in other words, by acting on each actuator in order to achieve the correct number of rotations. However, when applying an input command to each actuator, the LVDT indication did not match either its desired value or the real skin displacement.

Figure 3.1 shows the skin response (represented by a 'continuous line') after applying an input command to the first actuator. Measurements on both LVDT and on the skin are represented on this figure, as well as the desired input command. The graph shows that the measured LVDT indications are always smaller than the desired values. This offset between the actuator position (motor angle) and the 'LVDT indication' may be caused by the mechanical play inside the actuator mechanism and the difference between the desired and measured displacement on LVDT constitutes the compensation for this mechanical slack. The second issue is that the 'measured skin displacement' above the actuation point is also smaller to the desired displacement and the LVDT measured value. Possible explanations are that the wing structure (ribs, spars…) is not perfectly rigid but somewhat elastic and intern interactions between the skin and actuators; therefore, a displacement of 2 mm of the worm and wheel gears will not be entirely transmitted to the skin. Otherwise, the lower wing skin would be slightly changed.

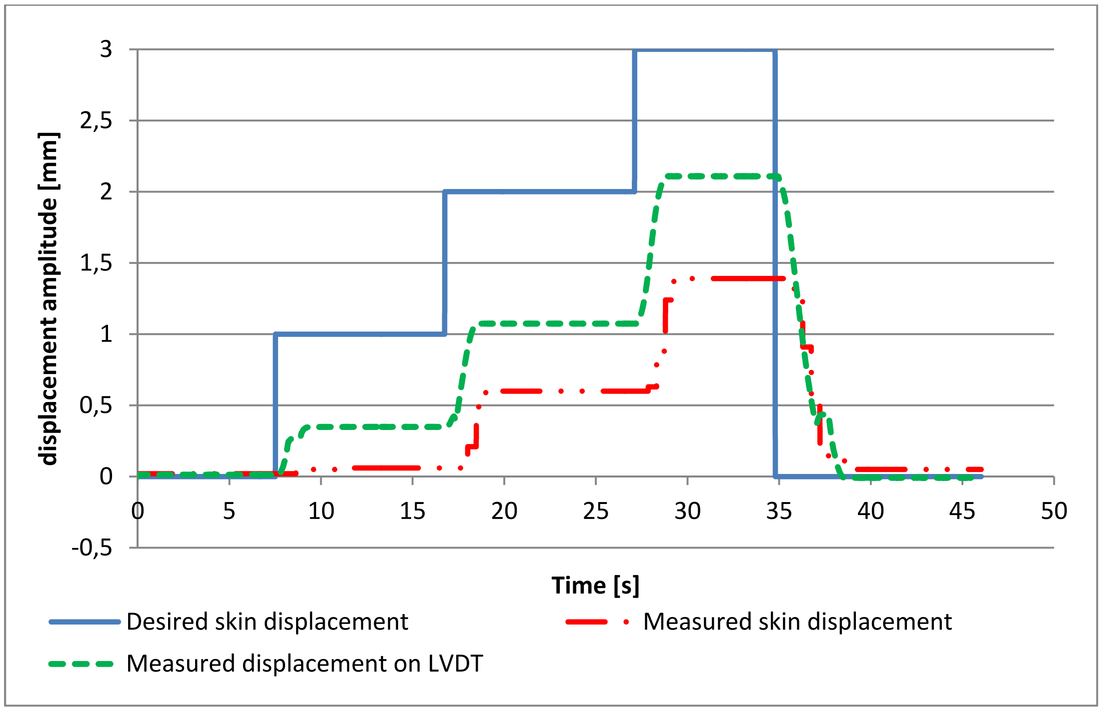


Figure 3.1 Skin response with open loop position control

Another issue is that the non-morphed position is not well defined and there is no reference that indicates the zero displacement point. In fact, when mounting the actuator inside the wing, each actuator should have its piston near the half-travel position and this position will be considered as zero displacement position. LVDTs should be placed so as to show almost zero mm of displacement; however, once we began moving the actuators, we noticed that the same input command with two different signs (e.g., +1 mm then -1 mm) did not bring the mechanism back to its initial position (based on LVDT indication). Also, since the motors are controlled by electronic drives, it is not possible to save the actual position of the motor when the drive is turned off or restarted (the actual position value will be lost). Given all of the above issues, we started thinking about using LVDT indication as a reference for the displacement of each actuator. In fact, unlike the electronic drive, LVDT has the advantage of retaining the measured value even when not excited (turned off), so it could be used as a reference for measuring the actuator stroke.

Noticing that there was a difference between the input command and the measured deflection of the skin, we tried to establish a correlation between the measured deflection of the skin and the indicated displacement value on LVDT. The idea was to represent the variation of the skin displacement versus LVDT indication in relationship to the zero deflection value of the skin. This representation would help to analyze and better understand the mechanical behavior of the entire system.

Table 3.1 Measurements (in mm) used to determine actuator offset values

	Actuator 1		**Actuator 2**		**Actuator 3**		**Actuator 4**	
Input	**LVDT**	**Skin dis**	**LVDT**	**Skin dis**	**LVDT**	**Skin dis**	**LVDT**	**Skin dis**
-1,00	-0,71	-0,59	-1,00	-0,70	-1,21	-0,68	-1,30	-0,65
-0,88	-0,59	-0,48	-0,90	-0,60	-1,10	-0,59	-1,17	-0,58
-0,75	-0,47	-0,38	-0,78	-0,50	-0,97	-0,49	-1,06	-0,47
-0,63	-0,35	-0,28	-0,67	-0,40	-0,91	-0,40	-0,94	-0,37
-0,50	-0,23	-0,18	-0,55	-0,30	-0,80	-0,30	-0,81	-0,27
-0,38	-0,09	-0,10	-0,42	-0,20	-0,69	-0,20	-0,69	-0,18
-0,25	0,11	-0,05	-0,29	-0,10	-0,57	-0,11	-0,55	-0,08
-0,13	0,12	-0,04	-0,12	-0,07	-0,40	-0,02	-0,39	-0,01
0,00	0,12	-0,04	-0,07	-0,07	-0,22	0,00	-0,23	0,00
0,13	0,23	-0,03	-0,04	-0,07	-0,15	0,01	-0,16	0,00
0,25	0,34	-0,01	0,00	-0,03	-0,09	0,01	-0,08	0,00
0,38	0,41	0,06	0,07	0,00	-0,05	0,02	-0,01	0,02
0,50	0,50	0,15	0,17	0,07	0,10	0,06	0,04	0,05
0,63	0,59	0,24	0,28	0,17	0,23	0,11	0,17	0,14
0,75	0,69	0,34	0,39	0,27	0,35	0,18	0,28	0,22
0,88	0,80	0,44	0,52	0,37	0,48	0,27	0,39	0,32
1,00	0,91	0,54	0,66	0,48	0,59	0,36	0,496	0,42

Clicours.com

The operation consists of separately moving each actuator with a small step between -1 mm and +1 mm, then measuring the skin displacement and taking the LVDT indication for each point. The Table 3.1 shows the results of the experiment for each actuator. These measurements will be used to define the 'dead zone' for each actuator. Figure 3.2 shows he results obtained for the first actuator, similar shapes are obtained with the other actuators.

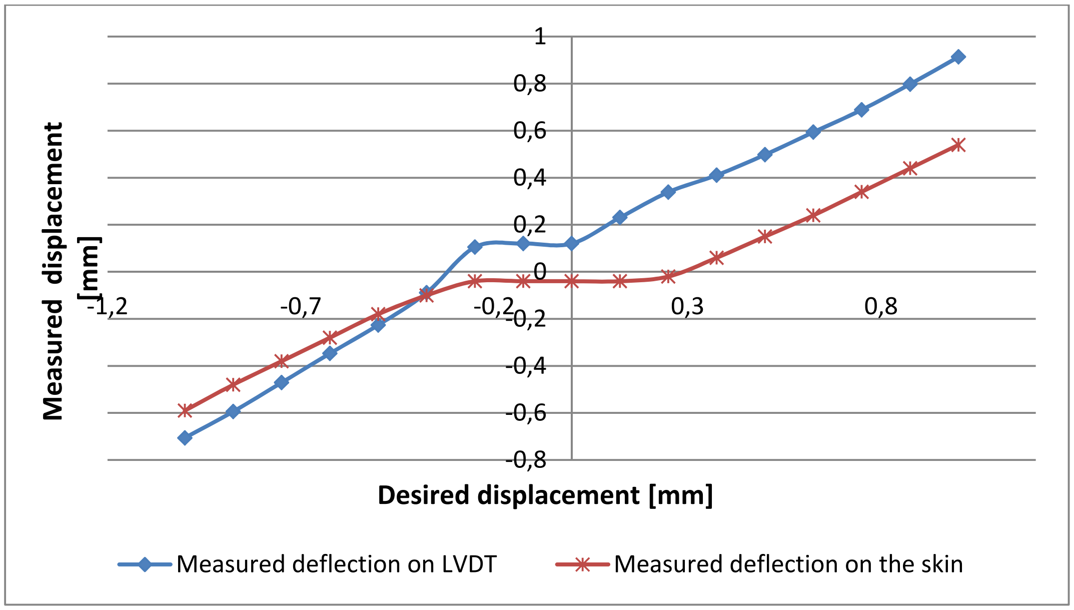


Figure 3.2 Actuator '1' displacement profile

In the above graph, the measured deflection is represented both on the LVDT and on the digital dial indicator (skin displacement measurement) for the actuator '1'. The curve with square marks, which represents the measured deflection on LVDT versus the desired input, shows a nonlinear behavior of the mechanism, while the graph becomes linear outside this zone. We can also see that the skin does not move for the steps between -0.25 mm and 0.3 mm on the desired input (curve with cross marks). The same type of behavior is observed for the actuators '2', '3' & '4' (see APPENDIX VII).

Since the LVDT indication is the only reference that will be used for displacement control, the measured real skin displacement is plotted against LVDT indication (Figure 3.3) to determine the ‘dead zone’ amplitude in terms of LVDT measurement variation.

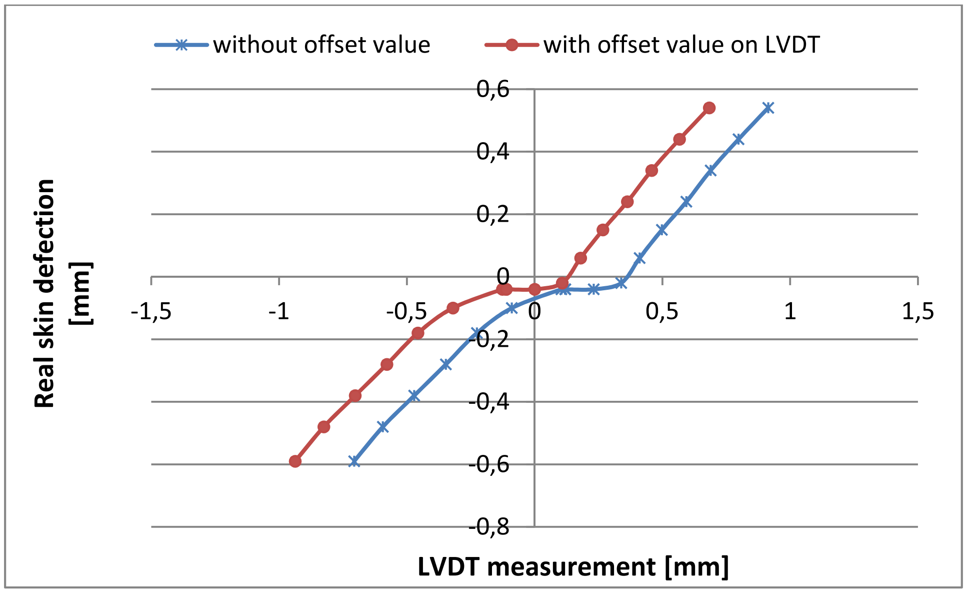


Figure 3.3 Skin deflection in terms of measured deflection on LVDT

As we can see, the curve (with cross marks) shows “a dead zone nonlinearity”; in other words, the skin doesn’t move during a certain range of values on the LVDT, indicating that the actuator is rotating but not producing the desired effect on the skin. Outside this range, the actuator is tight to the skin and the graph becomes almost linear. The ‘middle of the dead zone’ will be considered as a reference position and an offset value (half of the dead zone amplitude) will be considered for each LVDT channel; in this way, our reference position will be 0 mm of displacement on each LVDT. For this value, actuator mechanisms are in the middle of the ‘dead zone, ensuring that the skin is not morphed at all (graph with circle marks). The zero mm on LVDT should correspond to a null displacement value of the skin, which is not the case in the graphic (Figure 3.3); the problem is just a zero set issue on the digital dial indicator.

The same operation was performed using the remaining actuators by allowing us to resolve some of the noted critical issues. This preliminary “calibration” is very important at this step and will help us to achieve our primary goal of morphing the wing. The next step is to identify how to control the wing based on the measurement feedback from the LVDT value, since this measurement is the absolute reference for defining the real shape of the flexible skin.

3.2 Skin shape closed loop control

3.2.1 Principle of the closed loop

At the beginning of our investigation, actuator motion was controlled by acting directly on the motor position through the electronic drive. Since the control operation was not efficient in terms of precision and reliability, we decided to use the position feedback from the LVDT as a reference for the control operation.

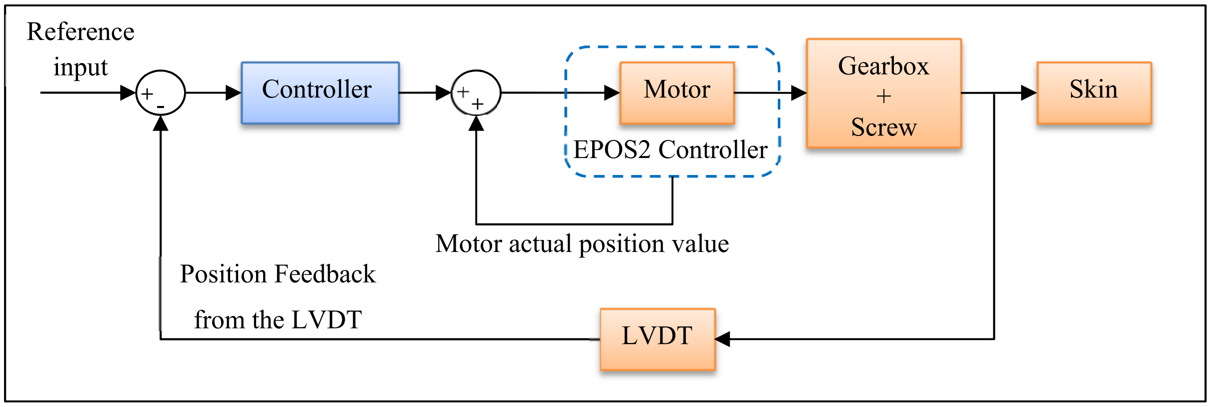


Figure 3.4 Principle of the position closed loop based on the LVDT feedback

The experiment shown in section 3.1.1 demonstrated that the flexible skin displacement above each actuator outside of a certain range is proportional to the indication of the LVDT, which is why a closed loop with feedback from LVDT will be essential to manipulating the skin shape, since the set point on the LVDT will subsequently be considered for the skin position control.

The closed loop is characterized by the feedback signal from the linear variable differential transformer; this feedback signal monitors the piston travel of the actuator and returns the measured value. The difference between the reference input and measured piston position will be the input of the controller, which calculates the correct position setpoint value that should be added to the actual position of the motor. The sum (Controller output + Motor actual position value) is then injected into the drive as a new set point for the motor position on the flexible skin (see Figure 3.4). In this way the motor will move until the piston reaches the desired position and overcomes the problem of mechanical play, which is not well quantified, but which will be compensated for each actuator.

3.2.2 Closed loop implementation:

Control decisions are handled by the real-time embedded system (NI PXIe-8135), based on the control model structure and feedback received from the hardware under control. Data flow diagram of the closed loop is shown on the Figure 3.5.

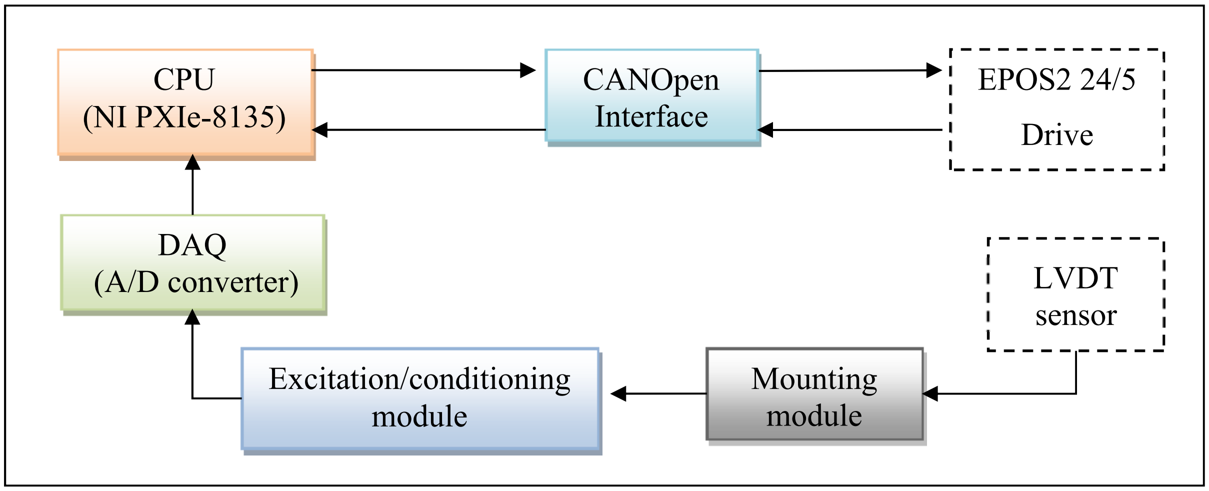


Figure 3.5 Data flow diagram for the closed loop control position on the LVDT

Feedback is received from the LVDT analog sensor mounted to an excitation and conditioning module; the sensor measurement is read via a data acquisition module (A/D converter). Feedback is also received from the brushless DC motor drive (EPOS2); the motor angular position is sent in real time via the CANopen communication protocol. In return, the real-time embedded system, handles all the communication tasks, receives feedback, calculates the control output, and acts on the motor drive by sending a new set point value.

3.2.3 Closed loop model structure and execution

The control architecture (Figure 3.6) is implemented through a MATLAB Simulink model, supported by the NI VeriStand environment. The model operates at a fixed frequency, which is the target operating system execution frequency (100 Hz). Each input of this model is updated at the same rate as the model's execution, while the drive command is actualized during each execution cycle (100 cycles per second). During each cycle, the input values are read and then the controller reacts to the actuator drive by recalculating and adjusting the input command. The plant (an Epos2 24/5 drive) responds to this change and another sample is taken. The controller only stops making changes when the actuator position reaches the set point.

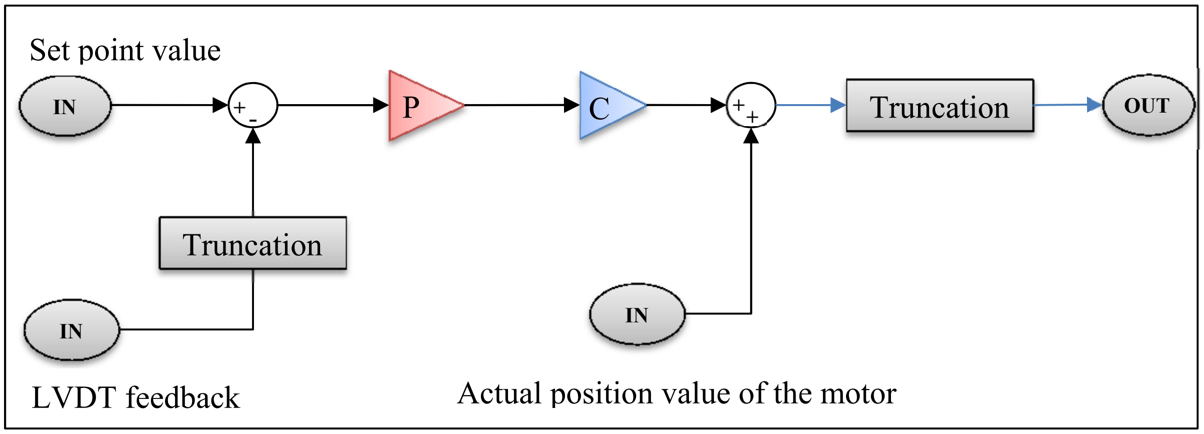


Figure 3.6 Position control model structure of the closed loop on the LVDT

The model has three inputs (IN) illustrated in Figure 3.6, inputs are:

- "LVDT feedback": This input receives the measurement value (in mm) from the LVDT sensor. Since the returned value is too precise (4 or 5 digits after the decimal point), a block for truncation is used to round out the value to double-digit precision (e.g., truncation 1, 23456~ 1, 25).
- The actual position value of the motor: The angular position value of the motor can be retrieved via CANopen communication; this value is returned in Maxon units qc (1 motor rotation is equal to 24 qc).
- Set point value: This input is the desired displacement value entered by the operator and should be given in mm.

And an output (OUT)

- Drive input: This output will be the input of the motor drive as it produces the new set point value for the actuator and should be in Maxon units (qc).

The difference between the set point value and the truncated measured value from the LVDT sensor is multiplied by the controller (Proportional) and then by the fixed gain in order to convert the units from millimeters to 'qc'. This difference value (in mm) is added to the actual position value retrieved from the smart drive via the CANOpen network and is then passed through a truncation block as shown in Figure 3.6. The choice of the controller as well as that of the detailed position loop architecture is explained in the next sub-section.

3.2.4 Adjustment of the closed loop parameters

The idea behind the design of the position control loop was to benefit from the internal control architecture of the EPOS2 drive, which offers high performances in terms of stability and dynamism. We programmed the drive to operate in 'position mode' (see section 2.2.3 for further details), adjusting the maximal speed and acceleration values so as to obtain a dynamic actuator response with a smooth position profile and tight displacement precision.

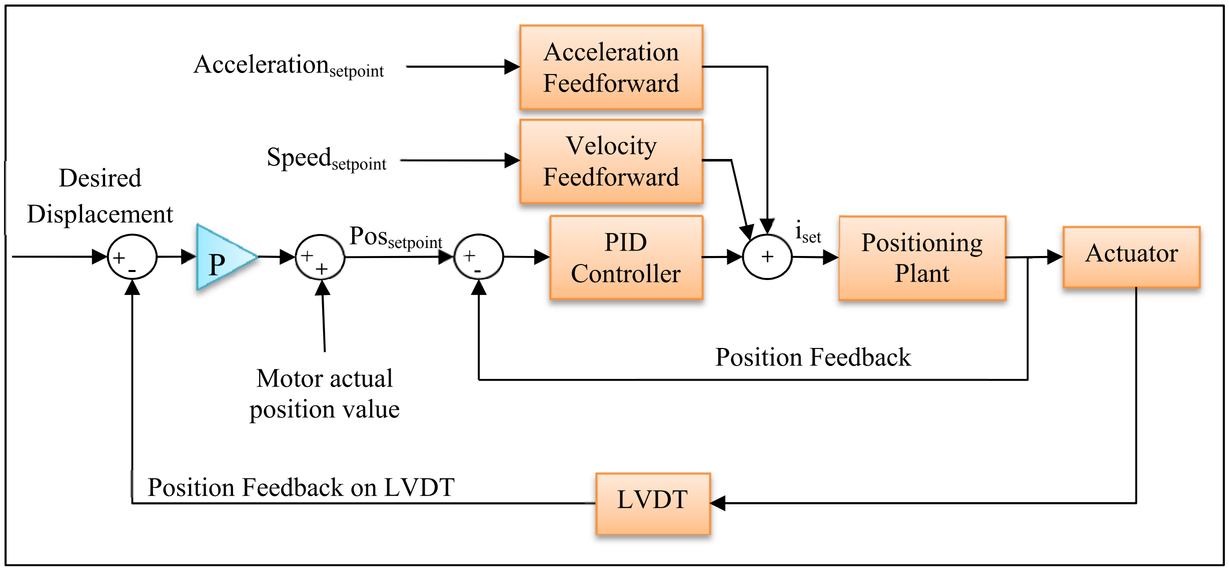


Figure 3.7 Position control loop detailed architecture

The internal position control loop (Figure 3.7) is implemented as a PID controller based on a subordinated current control (not shown here). Feedforward gains are added to improve the position set point tracking. Velocity feedforward is added to compensate for speed-proportional friction, whereas feedforward acceleration is used to compensate for the inertia effect.

The intelligent tool “Regulation tuning” of the EPOS2 software package was used to regulate the PID feedforward gains. The working principle of this tool can be summarized in three steps (EPOS2 application notes):

- Identification and modeling of the plant;
- Mapping model parameters of the plant to derivate controller parameters (PI, PID, and feedforward); and
- Verification of the resulting regulation structure.

A simple closed loop on the LVDT will not be entirely sufficient for actuator displacement control. In fact, the actuator displacement response which could be determined through the

LVDT response has revealed that there is a significant mechanical latency present in the system. In the Figure 3.8, we represent the response of the BLDC motor versus time, superimposed on the actuator response determined by the LVDT response. A delay of 350 milliseconds can be seen between the two plants. This difference is the time needed to move the piston tip from where it was when the gear box was actuated by the motor shaft.

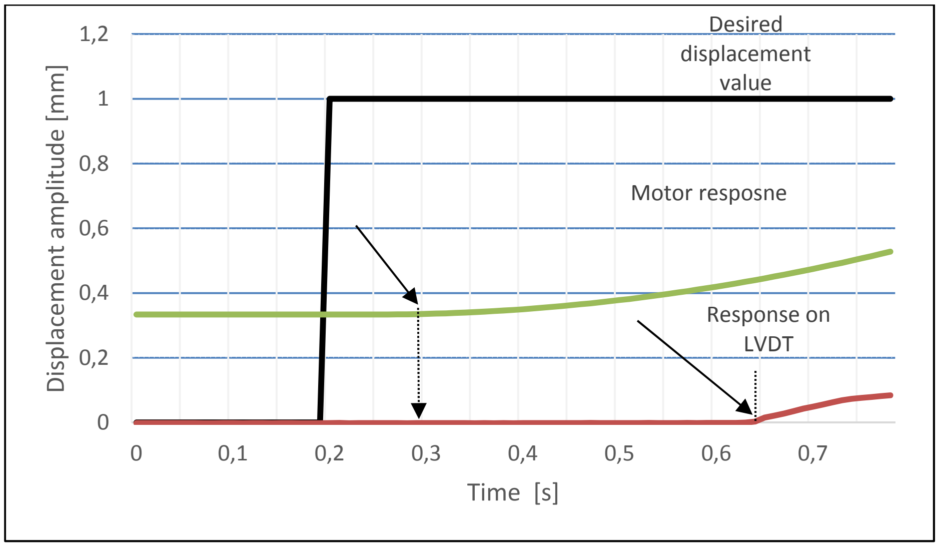


Figure 3.8 Mechanical latency of the actuator

With a simple closed loop (P = 1), the response presented an overshoot for a step of 1 mm amplitude, understandable because of the time delay between the BLDC motor and entire actuator response. We varied the proportional controller in order to find the coefficient that gives the optimal response, as shown in the figure below (Figure 3.9).

The best response was obtained with “Kp= 0.8”; this coefficient value reduced the amount of energy injected in the closed position loop, which gave the system enough time to adjust itself, thereby allowing the motion to stabilize and reach the desired position with a very small margin of error compared to the desired value (0.01 mm of static error).

The unusual shape of the actuator response seen on this curve (red oval) is due to the mechanical play between the different components of the actuators; the motor was rotating while the LVDT value was stagnant, which means that the piston was not moving during this period.

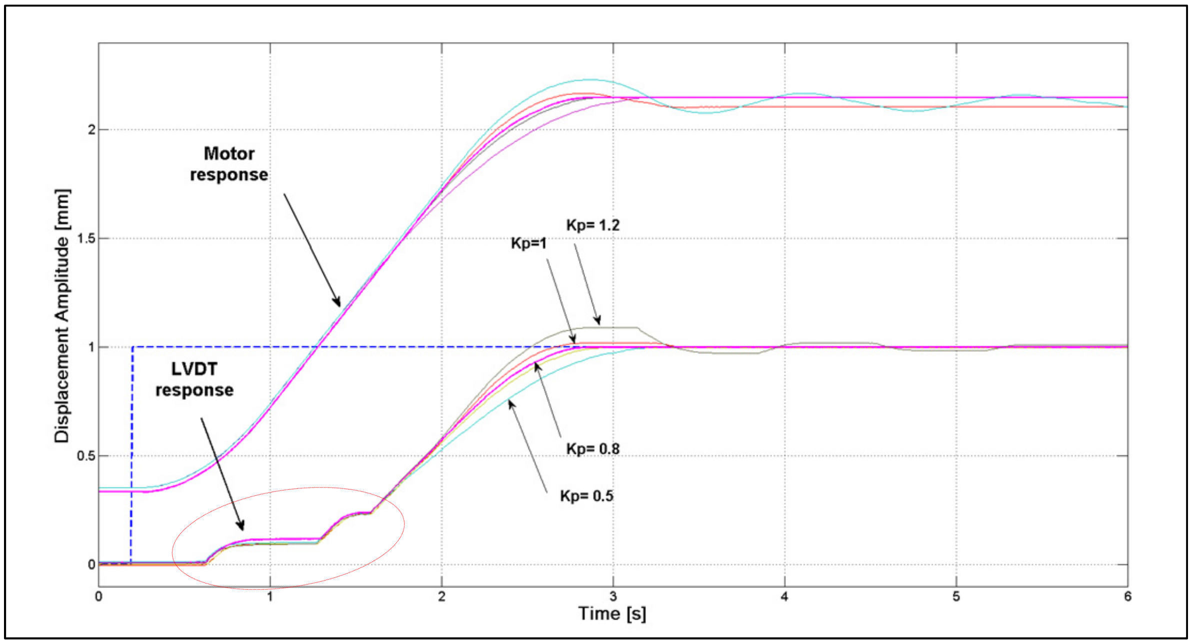


Figure 3.9 Actuator responses for different proportional gains

3.3 Skin shape morphing using lookup tables

3.3.1 Principle of 'lookup tables':

As shown in the previous paragraph, determining the real skin displacement from the measured value on LVDT is problematic, so it will not be possible to determine straightforward the input command to apply in order to reach a desired position on the skin.

Lookup tables are a kind of corrections table that will produce an approximate value for the required set point on LVDT value for a specified skin displacement value, as we can see in Figure 3.10. Several measurement points are taken in the skin displacement range. For each point, the value for skin displacement (measured through digital dial indicators) and its corresponding value indicated by the LVDT sensor are noted. Points are then linearly interpolated in order to entirely cover the displacement range.

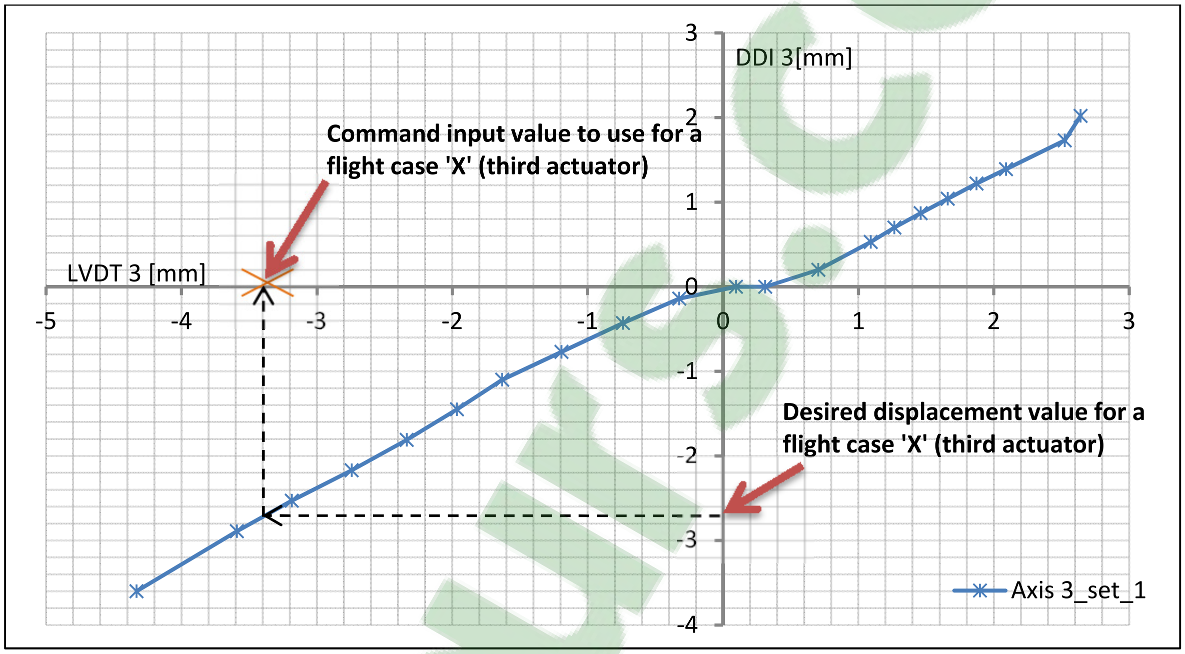


Figure 3.10 Principle of Lookup table

3.3.2 Open loop measurements

In order to set correction tables, we collected two sets of measurements. The difference between the two sets is the way in which the actuators are moved. Strategies were selected for data collection following some brief analyses of theoretical displacement values. Knowing that there is no offset value on the LVDT indications, we used existing values of LVDTs. However, we ensured from the beginning of the test that each actuator was located in the ‘mechanical play zone’ (Thus, if the motor turns in both directions with a small step (0.2 or 0.3 mm), no variation on the LVDT measurement or deflection on the skin is observed.

clicours.com

3.3.2.1 First measurement set

For the first set of measurements, the strategy was the following:

- The first and third actuators were moved until obtaining 2 mm of skin displacement (not LVDT indication) measured above each actuation point.
- The second and fourth actuators were moved until obtaining -2 mm of skin displacement measured above each actuator.
- We started simultaneously moving the four actuators with a step command of 0.4 mm as follows:

  -Commands on the first and third actuators were decreasing (from a positive displacement value toward zero).

  -Commands on the second and fourth actuators were increasing (from a negative displacement value toward zero).

Table 3.2 First measurement set

Input command in mm	DDI 1	LVDT 1	DDI 2	LVDT 2	DDI 3	LVDT 3	DDI 4	DDI 4
Initial point	0.00	0.09	-0.02	0.56	-0.02	0.05	-0.01	-0.01
Starting point	2.00	2.96	-2.00	-2.17	2.02	2.64	-2.00	-2.59
0.4	1.73	2.64	-1.73	-1.83	1.73	2.52	-1.78	-2.32
0.4	1.20	2.06	-1.18	-1.24	1.22	1.87	-1.25	-1.61
0.4	0.85	1.60	-0.82	-0.88	0.87	1.46	-0.91	-1.26
0.2 for 1&3/ 0.4 for 2&4	0.69	1.38	-0.64	-0.68	0.70	1.27	-0.74	-1.11
0.4 for 1&3/ 0.2 for 2&4	0.53	1.16	-0.29	-0.26	0.53	1.09	-0.42	-0.76
0.4	0.24	0.78	-0.12	-0.03	0.20	0.71	-0.26	-0.55
0.4	0.02	0.57	-0.03	0.40	0.00	0.31	-0.03	-0.03
0.4	0.03	0.54	0.00	0.61	0.00	0.10	-0.01	0.30
0.4	0.02	0.10	0.21	0.87	-0.14	-0.32	0.20	0.48
0.4	-0.22	-0.42	0.56	1.30	-0.43	-0.74	0.53	0.79
0.4	-0.54	-0.83	0.92	1.76	-0.77	-1.19	0.87	1.16
0.4	-0.86	-1.21	1.28	2.16	-1.10	-1.63	1.22	1.65

Input command in mm	DDI 1	LVDT 1	DDI 2	LVDT 2	DDI 3	LVDT 3	DDI 4	DDI 4
0.4	-1.20	-1.63	1.64	2.52	-1.45	-1.97	1.56	2.12
0.4	-1.54	-2.05	2.00	2.91	-1.81	-2.34	1.91	2.49
0.4	-1.89	-2.45	2.36	3.34	-2.17	-2.74	2.26	2.82
0.4	-2.25	-2.81	2.72	3.78	-2.53	-3.19	2.59	3.22
0.4 only for 1&3	-2.61	-3.19	3.07	4.19	-2.89	-3.59	2.92	3.69

3.3.2.2 Second measurement set

For the second set of measurements, the strategy was the following:

- The first and third actuators were moved until obtaining -4 mm of skin displacement measured above each actuation point.
- The second and fourth actuators were moved until obtaining 3.2 mm of skin displacement measured above each actuator.
- We start simultaneously moving all the actuators with a step of 0.5 mm (motor command):

  -Commands on the second and fourth actuators were decreasing (from a positive displacement toward zero mm of displacement).

  -Commands on the first and third actuators were increasing (from a negative displacement toward zero mm of displacement).

Table 3.3 Second measurement set

Input command in mm	DDI 1	LVDT 1	DDI 2	LVDT 2	DDI 3	LVDT 3	DDI 4	DDI 4
Starting point	-0.01	0.031	-0.02	0.531	-0.02	-0.009	-0.02	0.2
Intermediate point	-1.43	-1.919	1.16	2.118	-1.33	-1.897	1.07	1.579
Extreme point	-3.99	-4.71	3.37	4.6	-3.95	-4.727	3.2	4.178
0.5	-3.71	-4.6	3.12	4.361	-3.64	-4.506	3.01	3.937
0.5	-3.25	-4.12	2.68	3.908	-3.18	-4.052	2.59	3.355
0.5	-2.8	-3.581	2.25	3.376	-2.73	-3.602	2.18	2.844
0.5	-2.35	-3.075	1.81	2.845	-2.29	-3.105	1.75	2.447
0.5	-1.91	-2.628	1.38	2.385	-1.86	-2.625	1.33	1.954

Input command in mm	DDI 1	LVDT 1	DDI 2	LVDT 2	DDI 3	LVDT 3	DDI 4	DDI 4
0.5	-1.49	-2.176	0.94	1.936	-1.42	-2.147	0.92	1.389
0.5	-1.1	-1.61	0.51	1.405	-1.02	-1.702	0.5	0.883
0.5	-0.72	-1.066	0.08	0.873	-0.62	-1.205	0.1	0.48
0.5	-0.36	-0.606	-0.11	0.7	-0.24	-0.683	-0.07	0.398
0.5	-0.17	0.272	-0.45	-0.269	-0.09	0.193	-0.54	-0.747
0.5	0.11	0.684	-0.86	-0.825	0.27	0.697	-0.93	-1.146
0.5	0.53	1.198	-1.28	-1.286	0.73	1.191	-1.32	-1.582
0.5	0.93	1.721	-1.72	-1.726	1.14	1.613	-1.73	-2.134
0.5	1.34	2.284	-2.16	-2.254	1.53	2.099	-2.14	-2.653
0.5	1.74	2.737	-2.6	-2.787	1.93	2.556	-2.56	-3.06
0.5	2.13	3.169	-3.05	-3.228	2.33	2.892	-2.99	-3.496
0.5	2.52	3.699	-3.51	-3.651	2.72	3.132	-3.01	-3.495

3.3.3 Measurements analysis

The Figure below shows the first actuator profile (skin displacement = function (LVDT indication) for two sets of measurements. The difference between the two sets is basically the motion direction of two actuators relative to the other two actuators.

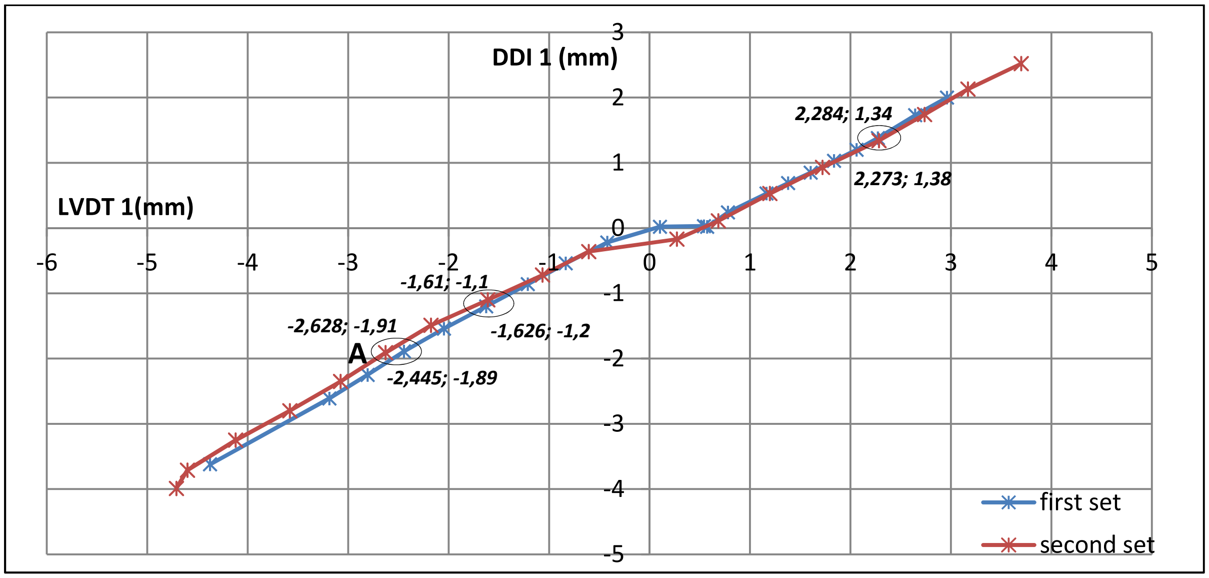


Figure 3.11 Skin displacement versus LVDT measurements for the actuator ‘1’

Linear interpolation is used in order to represent the discrete data table. Notice that:

- The curve is not linear;
- The curve represents hysteresis phenomena, which can be observed around 0 mm displacement value. (This can be seen outside the 0 mm zone less clearly, although the difference between the two curves is constant).
- A dead zone may be worth noting around 0 mm displacement value.
- For at least two different values of LVDT, the indication of the digital dial indicator number '1' is constant. For example, point 'A' is coordinated as [LVTD, DDI] = [-2.445, -1.89] mm on the first set, yet coordinated as [LVDT, DDI] = [-2.628, -1.91] mm on the second set. This behavior is not present in the positive displacement zone.
- The relationship between skin displacement and the LVDT indication could be considered to be linear outside the dead zone which is situated between [-1...1] mm of LVDT indication.

From the observations made above, we can say that it was not possible in this type of experiment to obtain a precise characterization of each actuator that would provide good results during the flight case validation step; however, we observe that the response of the skin above each actuator was slightly influenced by the displacement of the rest of the actuators as well as the configuration used for the experiment (the motion direction, actuators which are moving…)

3.4 Skin shape morphing using feedback on DDI

3.4.1 Principle of the closed loop on DDI

When attempting to morph the upper skin of the wing according to the optimized airfoils, the major problem was getting the four actuation points of the skin to the exact calculated displacements with very high precision. The aerodynamic team recommended that the error at each displacement point should not exceed 0.1 mm or efficiency could degrade. Since the displacement of each actuator was controlled by the set point on the LVDT closed loop, the

challenge was determining the correct value of a set point on LVDT that would bring the skin to the desired displacement.
The actuators were initially controlled using LVDT feedback and the skin displacement was simply measured at four points using the digital dial indicators (DDIs). Now, the position feedback for displacement control would no longer be retrieved from LVDTs but directly from the numerical indicators. The objective of this procedure was to achieve desired displacements in the absence of airflow, with the aim of repeating them during wind tunnel testing without using dial indicators; however, LVDT values of the optimized skin would be saved and used during the wind tunnel tests.
In order to achieve a very precise displacement on the skin, a 'position control loop' was implemented based on feedback from the digital dial indicators. Displacement values read through Universal Serial Bus (USB) were sent via User Datagram Protocol (UDP) to the real-time controller to be used for the position control operation as detailed in the next paragraph.

The closed loop is characterized by the feedback signal from the displacement sensor. The feedback signal will monitor the skin displacement and return the actual measured value. This value will be compared to the input reference, then the error signal representing the difference between the desired displacement value and the actual measured value will serve as the input for the controller. The controller will make the required corrections to reduce the error and bring the actuation point to the desired position.

Figure 3.12 shows the principle of the control loop. A proportional controller is used, so that the error between the measured and the desired displacement is multiplied by a proportional coefficient P, the control law is defined by the equation (3.1) as follows:

$$\textit{Drive setpoint} = P * (\textit{desired position} - \textit{current position}) + \textit{motor actual position} \qquad (3.1)$$

The output of the controller is then added to the actual position value of the motor in order to compensate for the difference between the desired and the effective displacement.

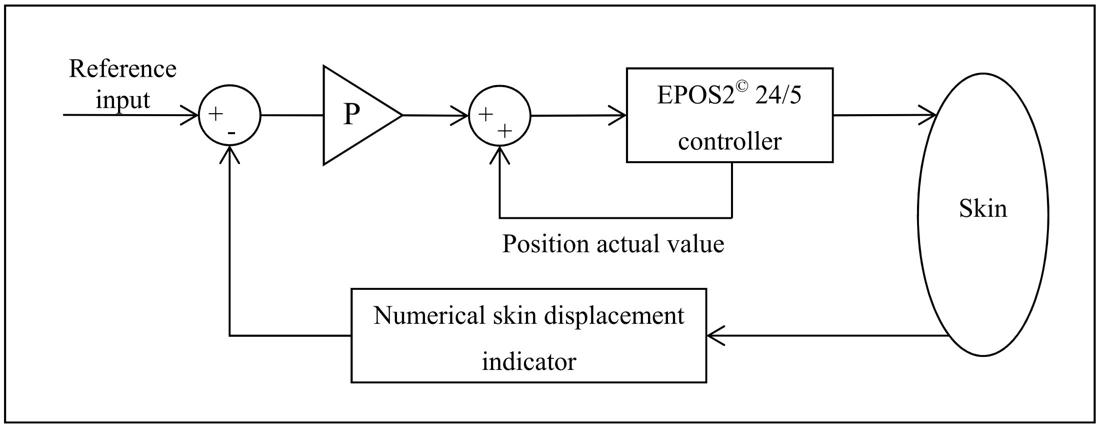


Figure 3.12 Position closed loop on digital dial indicators

3.4.2 Retrieving displacement values from DDI

The commercial solution (software) received with Mitutoyo® indicators offers the possibility of saving measured values to an Excel file, but such a solution does not allow us to implement the position control loop. These devices use a virtual serial communication protocol to make it possible to communicate using a serial communication port (COM) (Mitutoyo, 2013). A LabVIEW® program was developed to establish communication with the four electronic indicators, read the values and send them to the real-time controller (PXI-e) so that they may be used for the position control loop (Figure 3.13).

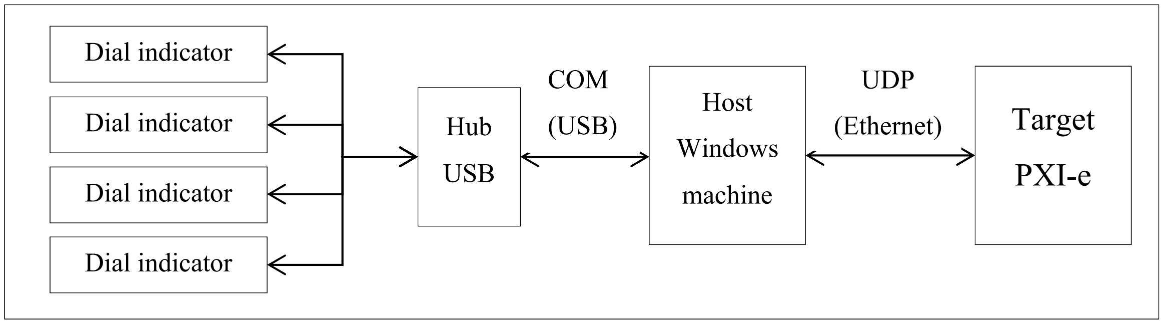


Figure 3.13 Data flow diagram between digital indicators and the PXI-e

The LabVIEW® program has two main functionalities. The first functionality is to establish USB communication between the host (Windows machine) and the four digital indicators, thereby allowing us to read the real displacement value of the skin. To implement this functionality in LabVIEW® we used the NI-VISA (Virtual Instrument Software Architecture) library. This library is a standard for programming and configuring instrumentations systems comprising GPIB, Ethernet, USB and other interfaces. VISA is the programming interface between such interfaces and the programming environment (LabVIEW®). The following is the procedure for establishing USB communication between the Windows machine and the digital indicators:

1) Open a session to the specified device.
2) Write the 'Read command' to the device.
3) Read the returned data from the device.
4) Close the device session.

The second functionality is to send retrieved values via UDP to the target. UDP is a basic low-level communication protocol between computers from the transport layer of the Internet protocol suite (known as TCP/IP)(Internet protocol suite, 2015). It is used for applications in which reliability is not critical.

In our case, we will use UDP protocol to send data from the Windows machine to the target and vice versa. When sending each datagram, the IP address of the destination and the port number should be specified. In LabVIEW®, the 'Protocols Vis and Functions' contains blocks that allow some UDP operations, such as opening a UDP socket on a specified port, reading a datagram from a UDP socket, writing to a remote UDP socket, and closing a UDP socket.

At this step, measurement values are continuously sent to the PXI and received at a defined UDP port. One issue was that the program didn't allow for all four measurements to be read at the same time. In fact, it works by opening a session with the first device via the first communication port (e.g., COM 1), then moving on to the second device, and so on. The time required to read all four values is between 300 and 500 milliseconds, so for the same measurement device, there is a variable delay that could reach a half of a second between each two successive measurements. The sent measurement data is recovered and rendered usable with a custom device.

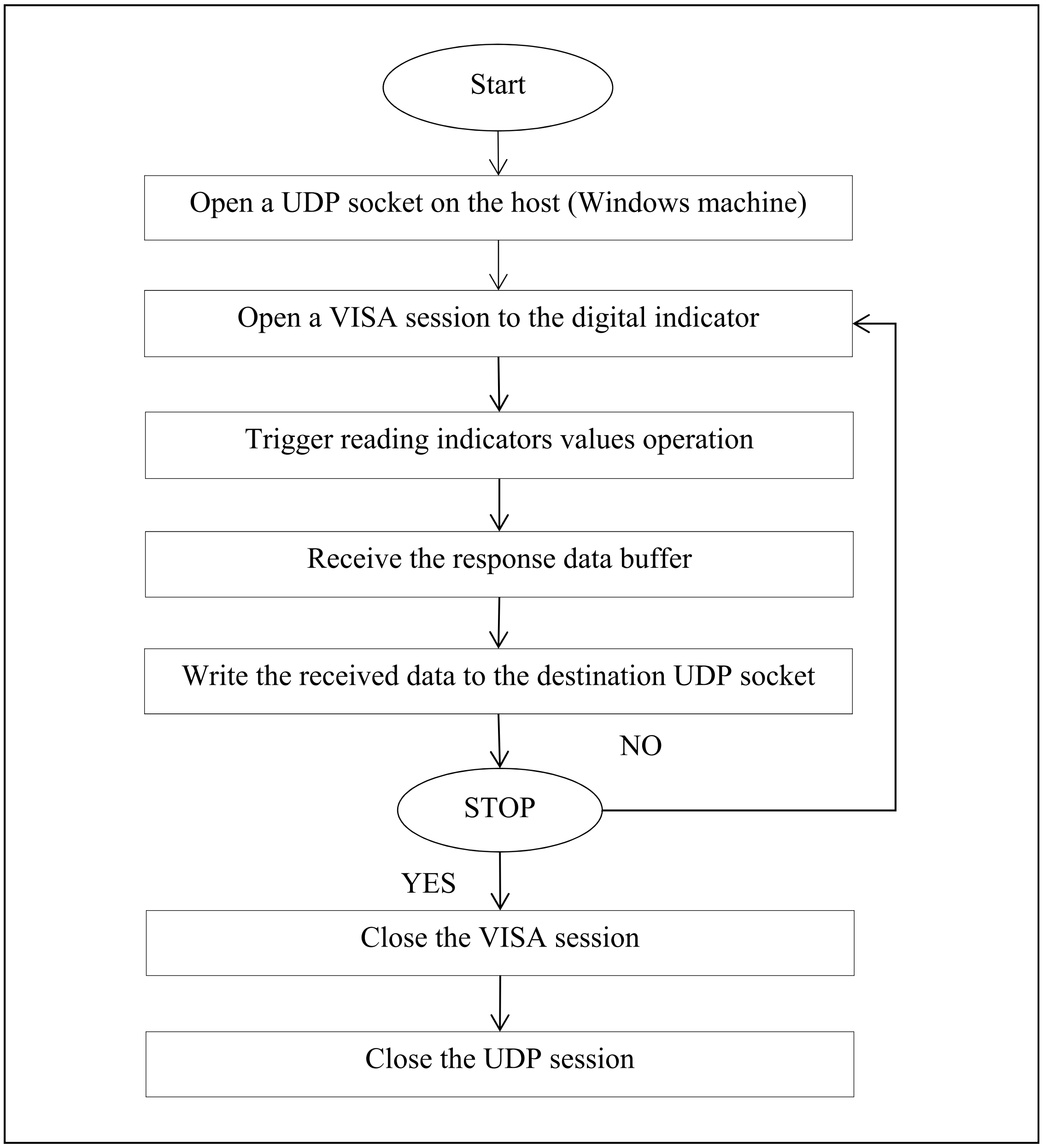

Start
Open a UDP socket on the host (Windows machine)
Open a VISA session to the digital indicator
Trigger reading indicators values operation
Receive the response data buffer
Write the received data to the destination UDP socket
NO
STOP
YES
Close the VISA session
Close the UDP session

Figure 3.14 Reading digital indicators values diagram

Measurement values are recovered at the output of the custom device module. Between each two successive measurement operations the last measurement value is maintained until the new reading value is received. This task was to be the basis of the skin displacement control operation, but so far we only had a reading of the skin displacement value with high precision and were not yet able to apply a desired value to the skin displacement. The next paragraph will present the solution adapted to solve this problem.

3.4.3 DDI closed loop implementation

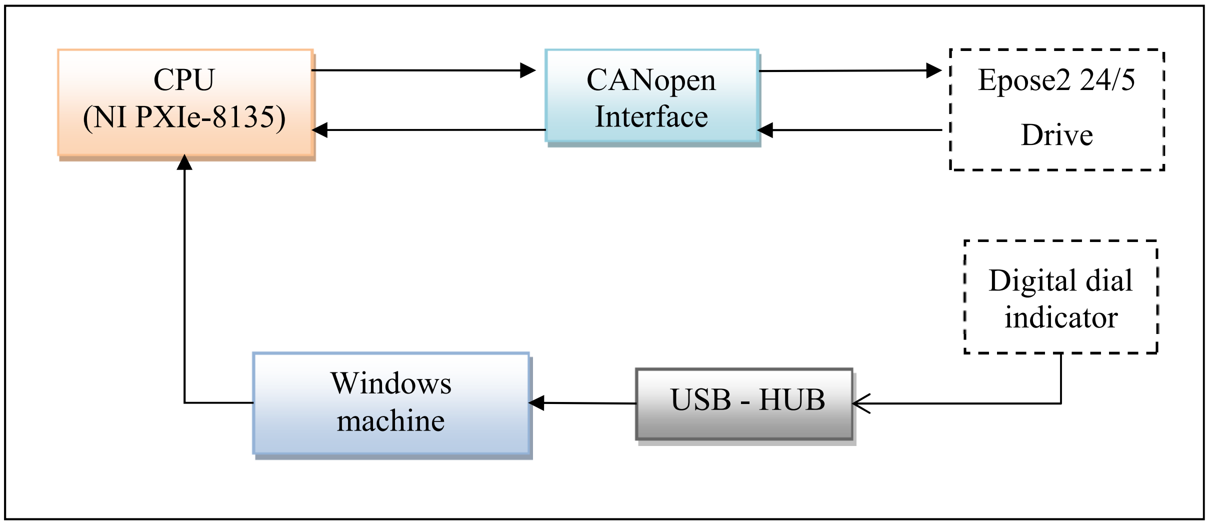


Figure 3.15 Data flow diagram for closed loop control position on LVDT

The hardware architecture used for implementing the closed loop on digital dial indicators feedback is shown in Figure 3.16. The motion components are the same as in the closed loop on LVDTs; changes affect only the position feedback components since we are concerned with the real skin displacement measured through DDI. Details about communication between the real-time controller and DDI were presented in the previous paragraph.

3.4.3.1 Closed loop model structure and execution

As seen in 3.2.3, the Simulink model operates at a fixed frequency, which is the target operating system execution frequency (100 Hz). All inputs and outputs of this model are updated at the same rate as the model's execution; at each execution, input values are received and the controller acts on the actuator drive by varying the output command.

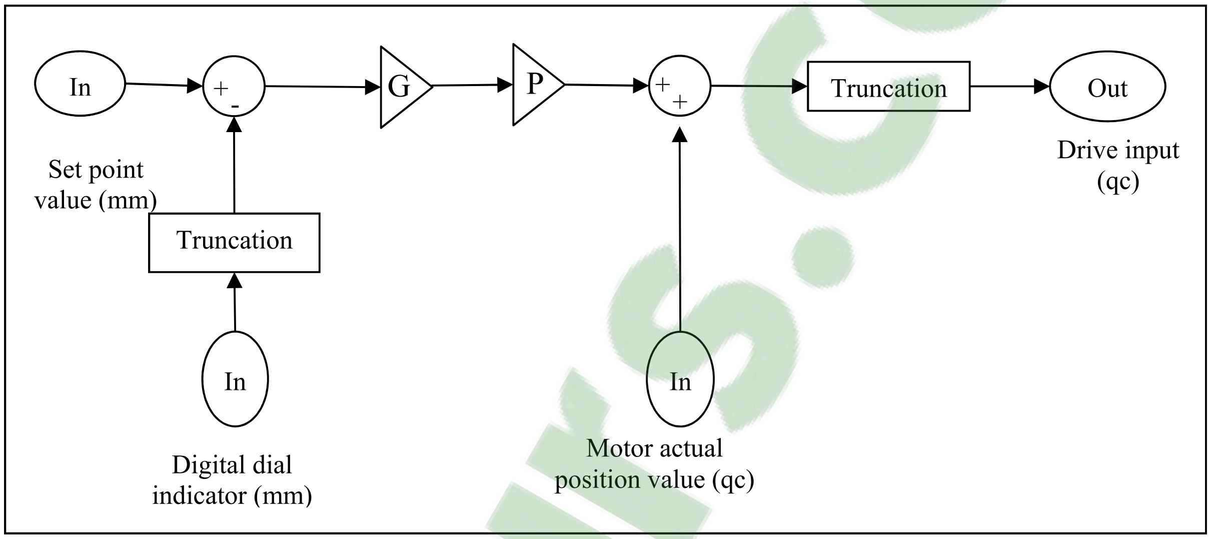


Figure 3.16 Position control model structure of the DDI closed loop

As seen in Figure 3.16, this model has three inputs:

- Digital dial indicator: This input receives the measurement value from the LVDT sensor in mm. Since the returned value is too precise (4 or 5 digits after the decimal point), a block for truncation is used in order to round out the value to double-digit precision (e.g., truncation (1, 23456) ~ 1, 25).
- Actual position value of the motor: The angular position value of the motor can be retrieved via CANopen communication; this value is returned in maxon units qc (1 motor rotation is equal to 24 qc).
- Set point value: This input is the desired displacement value entered by the operator and should be in mm.

And an output:

- Drive input: This output will be the input of the motor drive as it produces the new set point value for the actuator and it should be in maxon units (qc).

Clicours.com

The difference between the set point value and the truncated measured value from the DDI is multiplied by the proportional controller coefficient ‘P’ and then by fixed gain ‘G’, which will convert the unit from mm to maxon units (qc), as seen previously.
This value is added to the actual position value retrieved from the smart drive via the CANopen network and then passes through a truncation block as shown in Figure 3.16.

3.4.3.2 Adjusting the calibration closed loop parameters

The controller will attempt to reduce the amplitude of error so that the input of the drive will vary at a low rate. The fact that the drive input varies smoothly will result in the motor moving at a moderate speed so that the desired position will be achieved with very high precision.

The speed command of motors is varied by the EPOS2 drives depending on the position input command. In fact, the speed profile is established based on the amplitude of the displacement; the drives will calculate acceleration and deceleration rates to reach the desired position as soon as possible.

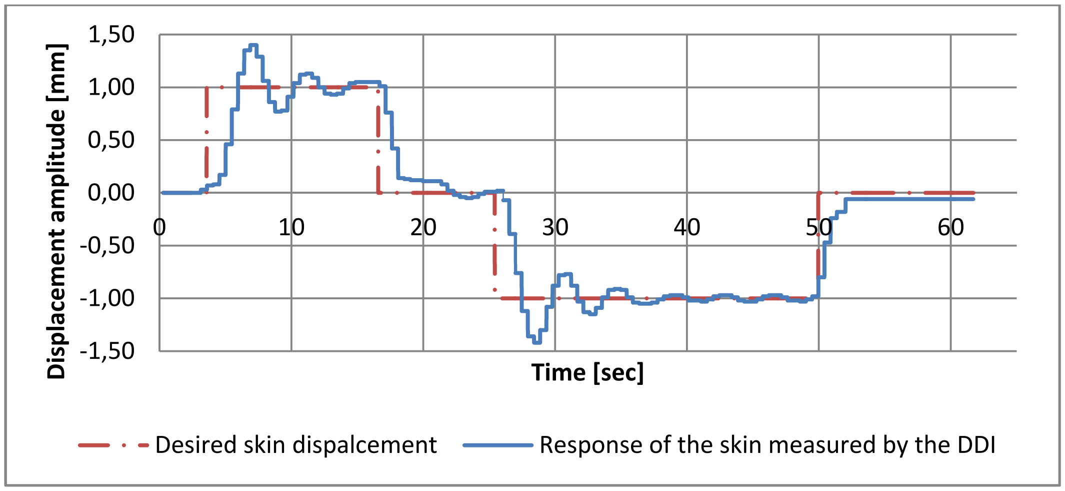


Figure 3.17 Skin response with closed loop on digital indicator without taking into account the proportional controller

Without the controller, the amplitude of error handled by the drives is significant, thus acceleration and deceleration rates are also significant for achieving displacement within a brief period of time.

In addition, there is a delay in updating the measured value returned by the numeric dial indicator, making it difficult to end the motion at the correct displacement position; motion will always exceed the desired value and start oscillating around the set point, while the control operation will end with a static error of 0.1 mm as shown in the Figure 3.17. The coefficient of the proportional controller was tuned experimentally to produce the best results in terms of precision and efficiency. A trial and error method was adopted, characterized by many attempts with a varied proportional coefficient until overshooting was eliminated (Figure 3.18); as a result, static error was also eliminated (it was found to be smaller than 0.03 mm).

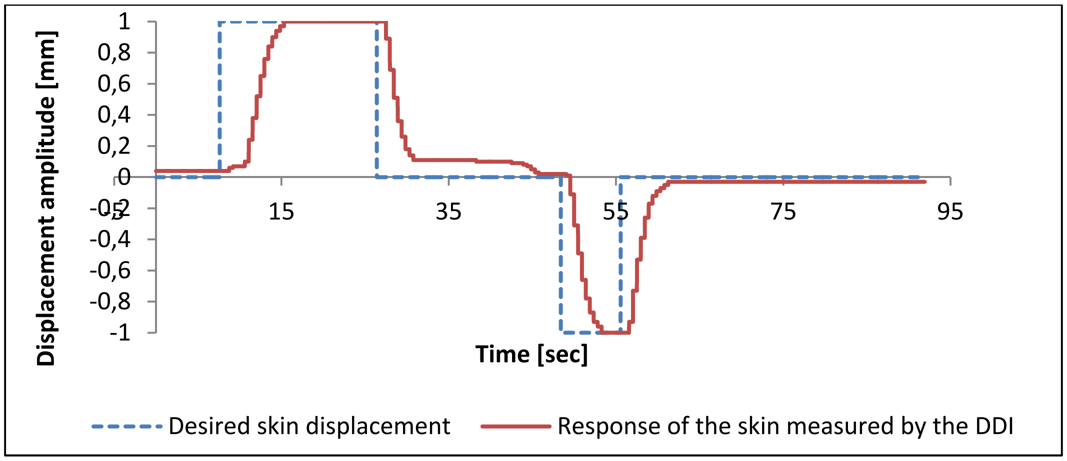


Figure 3.18 Response of the closed loop after adding the proportional controller

3.4.4 Optimization of the wing shape for one flight condition

After achieving a precise closed control loop on the first actuator, the same control strategy was used for the rest of the actuators. The next step was to optimize the wing shape for one flight case; in other words, we needed to obtain the desired displacement for each of the four actuators, and each actuator had to have a specific displacement value.

Displacement values are imported from a database that includes necessary information for each flight case. Data is arranged in a matrix with seven rows and a number of columns equal to the number of flight cases. Each column represents a flight case and contains four displacement values for each axis, wing angle of attack in degree (α_i), flow Mach number (Mach num_i) and aileron deflection angle (δ_i), in this order. Flight cases are called by number; necessary information is loaded and then distributed.

Table 3.4 Database organization

FC_1	**FC_2**	**....**	**FC_{n-1}**	**FC_n**
Δd_{11}	Δd_{21}	....	$\Delta d_{n-1,1}$	$\Delta d_{n,1}$
Δd_{12}	Δd_{22}	....	$\Delta d_{n-1,2}$	$\Delta d_{n,2}$
Δd_{13}	Δd_{23}	....	$\Delta d_{n-1,3}$	$\Delta d_{n,3}$
Δd_{14}	Δd_{24}	....	$\Delta d_{n-1,4}$	$\Delta d_{n,4}$
α_1	α_2	....	α_{n-1}	α_n
Mach num_1	Mach num_2	....	Mach num_{n-1}	Mach num_n
δ_1	δ_2	....	δ_{n-1}	δ_n

We noted previously that the measured skin displacement value of each actuator could be influenced by the movement of another actuator, especially when they are on the same chord line. Actuators on the same span position do not have any significant influence over each other. When we started moving the four actuators simultaneously, we found that some of the four actuators might not reach the exact value. One possible explanation is the mutual influence of actuators while in motion: when too near the final position, there is a big load on each actuator and motors are unable to resume moving once they stop. This always happens when one actuator reaches the end of its stroke while another is still underway.

To avoid this kind of problem, the adapted solution was to move actuators two by two with a delay between each group. In fact, for all flight cases, actuators on the same span position always have the same motion direction; either both actuators are pushing the skin or both are pulling the skin. By treating them as groups of two, different motion directions can be achieved: the first and third actuators will begin motion and once they are in full motion (half of the final displacement value), the second and fourth actuators' motion is triggered. In this way, we can ensure that offset is added to each of the two groups when motion is triggered, producing increased precision.

When the upper skin is morphed as desired, indicated LVDT values will be saved and this will determine with very high precision the required set points for the position control loop with feedback on LVDTs for this flight case. In this way it will be possible to reproduce the shape of the skin without having to measure its displacement again. Rather, only the set points on LVDTs will be used, which are the only option for morphing the wing with airflow.

Although the closed loop on numeric dial indicators has solved the problem of the non-linearity of skin displacement response, we cannot consider that this solution is final as the number of flight cases is significant and could vary with the testing phase and results obtained. Also, during transportation and installation of the wing, LVDT indications for the same skin shape could change, distorting previous configurations, as there is a need to execute displacements for each flight case with very high precision. In order to avoid such a risk, a calibration procedure was designed and implemented.

3.5 Flight case calibration procedure

3.5.1 Principles of flight case calibration

The calibration procedure was developed to both reproduce theoretical displacements of the four actuators for each different flight case with high precision and reproduce displacements for all flight cases.

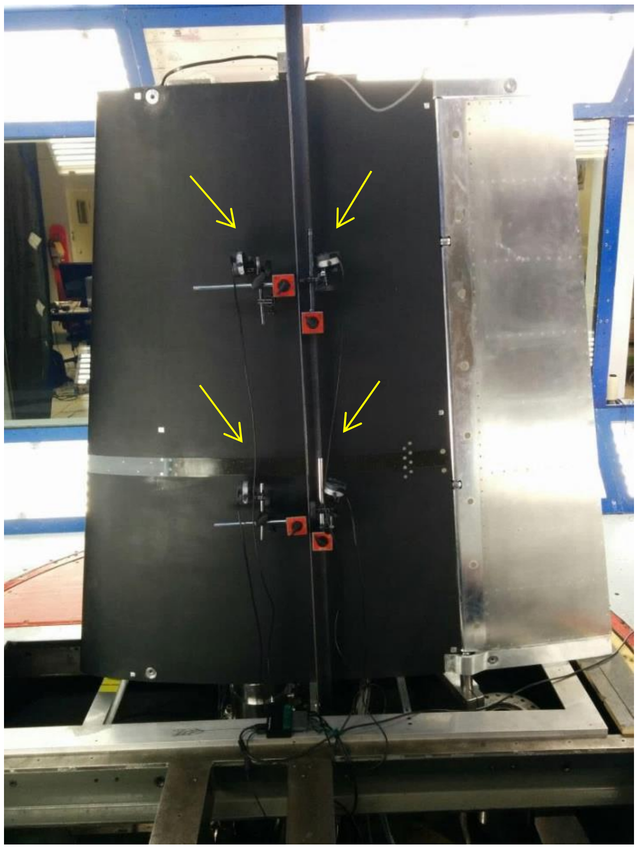


Figure 3.19 Digital Dial Indicators installed on the wing for calibration during WTTs

According to the feedback from the dial indicator, the four displacement values for each flight case should be reached. LVDT values corresponding to these displacements are logged and the output of calibration is saved under the form of a matrix that does not contain actual skin displacements, but it contains LVDT set point values allowing for real skin displacement values to be achieved.

The calibration process should be easy and quick so that it can be executed after installation of the wing in the wind tunnel, just before blowing. The execution of the calibration procedure after the wing installation will ensure excellent results in terms of their accuracy and precision. Calibration should be automated and of course not time consuming; the range of flight cases to calibrate is variable and could be specified before launching the operation, and all the selected flight cases are further executed sequentially. The calibration result should be quickly post-processed so that it may be used for morphing wing control during wind tunnel operations.

The Figure 3.19 shows the way in which DDIs were installed on the wing for calibration during the wind tunnel tests that took place in June 2015. A vertical metallic frame attached to the wing along the spanwise direction was used as a support on which magnetic bases were fixed then digital dial indicators were fixed orthogonally to the skin in such a way as to detect skin displacement.

3.5.2 Models' mutual interaction during flight cases calibration

During calibration, the main model in the integration architecture is the control model with digital dial indicator feedback, since our set point is on the skin displacement value and not on the LVTD value. In fact, the entire cycle is triggered by entering (or selecting) a flight case number; this step allows the loading of the desired displacement values from the database and sending them to the next corresponding model, which is the skin displacement model. When the desired values of the skin displacement appear at the input of the control model, motion starts and the skin moves until the response settles and the skin shape is optimized. Then, the motion is stopped.

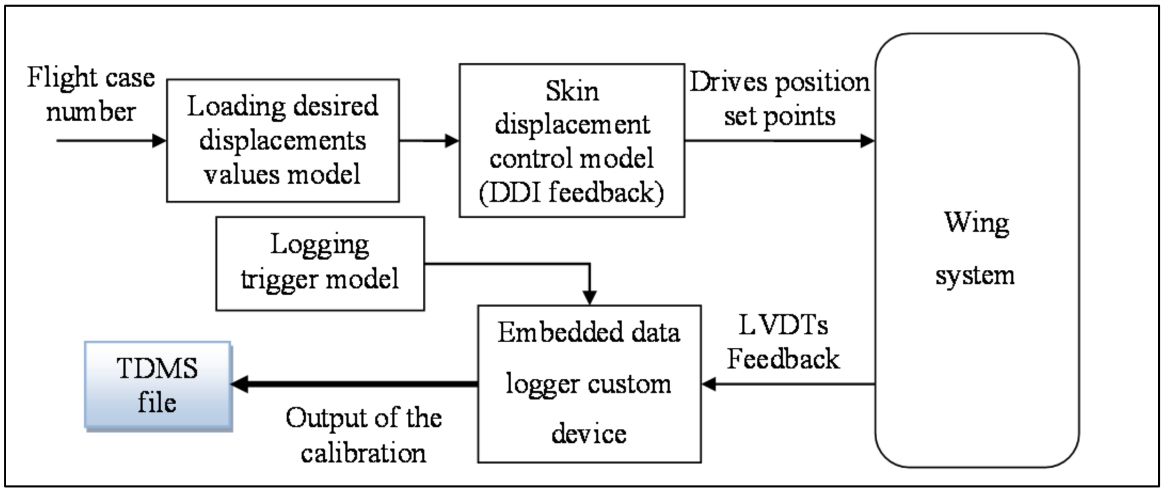

Flight case number
Loading desired displacements values model
Skin displacement control model (DDI feedback)
Drives position set points
Wing system
Logging trigger model
Embedded data logger custom device
LVDTs Feedback
TDMS file
Output of the calibration

Figure 3.20 Simplified models communication diagram

At this step, the operator has to trigger the embedded data logger custom device in order to log LVDT indications to a TDMS file. A post-processing step is required in order to extract the calibration data and transform it into a usable form.

The above diagram above illustrates the operation concept for one flight case calibration, but in practice the calibration is executed automatically for all cases one after the other one in order to optimize execution time and facilitate the operation.

3.5.3 Automating the calibration procedure:

Before launching the calibration, it is necessary to check if each actuator is at its reference position. In fact, we have previously demonstrated that a non-morphed skin shape corresponds to 0 mm on LVDT indications; therefore, after installing the digital dial indicators at their positions on the skin, it is necessary to check if all the actuators are at 0 mm of displacement (based on LVDT values) or if they need to return to zero. Since this position is the non-morphed skin shape, digital dial indicators must be set to zero.

After this preliminary step, the actual calibration procedure can be launched. The automation is implemented using the ‘Stimulus Profile’ tool from National Instruments by ensuring that all the control channels are controlled and monitored by the system following the implementation of a real-time sequence.

The diagram below (Figure 3.21) illustrates the logics behind the ‘automated calibration procedure’. The first step is to check the reference position. 0 mm on LVDT indication should correspond to 0 mm on the digital dial indicator for each of the four actuation points; otherwise, an error is returned and the operation is aborted. After the validation of the first step, the drives is enabled and the flight case number is incremented (the default value is 0, which represents the ‘non-morphed configuration’).

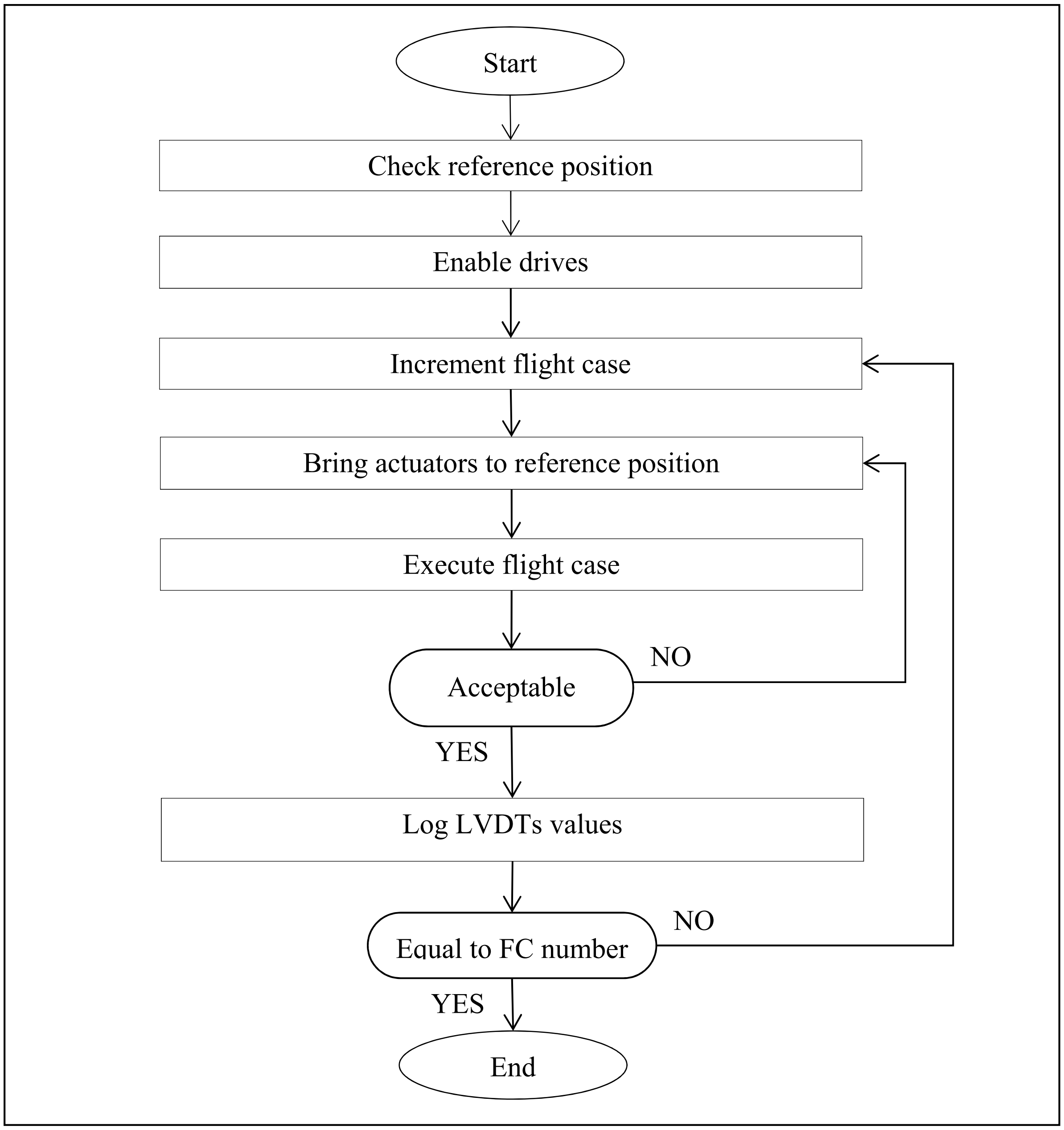


Figure 3.21 Calibration execution diagram

The system will take the needed time to execute the flight case and bring the skin to its optimized shape. If the error is acceptable (below 0.05 mm), the procedure keeps running; if not, actuators are brought back to their reference positions, and the flight case is restarted. Once the error obtained is below the threshold value, LVDT values are logged and the cycle restarted until all the flight cases are calibrated. Note that the operator can specify the number of flight cases or define a range of flight cases to execute.

3.6 Skin shape morphing using system identification

In the previous paragraph, we presented the calibration procedure that was the adapted solution for morphing the skin following optimized airfoils for each flight case. We then tried to use an artificial neural network (ANN) to identify the nonlinear response of the skin. In fact, a PhD student at LARCASE carried out some studies concerning artificial neural network learning (ANN) for identifying nonlinear system models. We collaborated in order to apply ANN as an alternative to the calibration procedure. We will only discuss the system integration aspect and data collection, as we did not work on the part related to the learning method or development of the ANN.

3.6.1 Principle

The idea behind system identification using ANN is to establish a relationship between skin displacement and LVDT indications that could be valid for the entire displacement range of the flight cases. We are not able to use DDI during wind tunnel tests thus the ANN system will be our alternative, as it will predict skin displacement values on the basis of LVDT readings in real time as shown on Figure 3.22

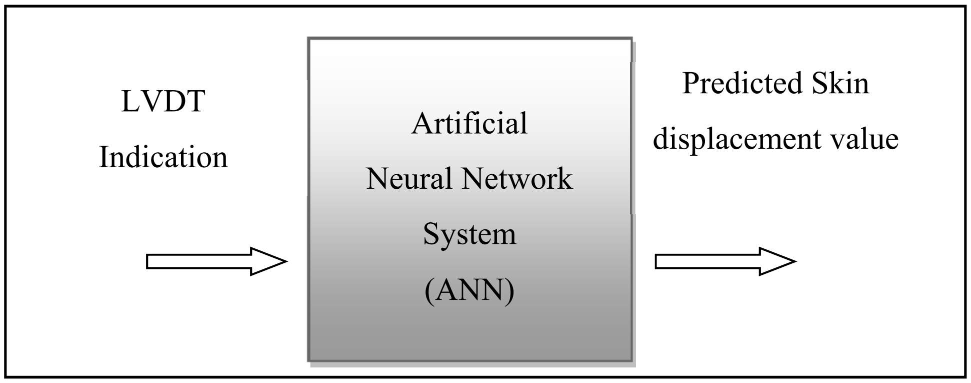


Figure 3.22 Principle of the artificial neural network system

3.6.2 Data collection

‘Data collection’ is a crucial step in the development process of the neural network system. As for the training step, this data will be used to establish a particular relationship within the network structure (layers of neurons, connections between neurons…) so that a particular input value steers to a specific output value.

Table 3.5 Training data organization

LVDT 1	**LVDT 2**	**LVDT 3**	**LVDT 4**	**DDI 1**	**DDI 2**	**DDI 3**	**DDI 4**
X_{11}	X_{12}	X_{13}	X_{14}	Y_{11}	Y_{12}	Y_{13}	Y_{14}
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
X_{n1}	X_{n2}	X_{n3}	X_{n4}	Y_{n1}	Y_{n2}	Y_{n3}	Y_{n4}

System configuration during data collection should match the wind tunnel tests (WTT) flight conditions as much as possible; otherwise, the final results of the learning procedure will be different compared to actual skin response. To collect required data points, actuators were moved so that they covered the entire data range of the flight cases, and calibration data for the first and second flight case sets was also used. Data takes the form of a large array of discrete points; for each point, we saved LVDT indications for all the actuators, as well as DDI indications above each actuation point on the skin, as seen in the table Table 3.5. The LVDT indications were denoted by X_{ij}, where ‘i’ is the number of flight case, and ‘j’ is the number of actuator (j=1, 2, 3, 4). The DDI indications were denoted by Y_{ij} where where ‘i’ is the number of flight case, and ‘j’ is the number of actuator, also.

3.6.3 Model integration and implementation

The ANN system was trained based on the data collected, as mentioned above. The next step, after testing and validation in simulation, is its integration to the control loop and testing of the skin’s final response.

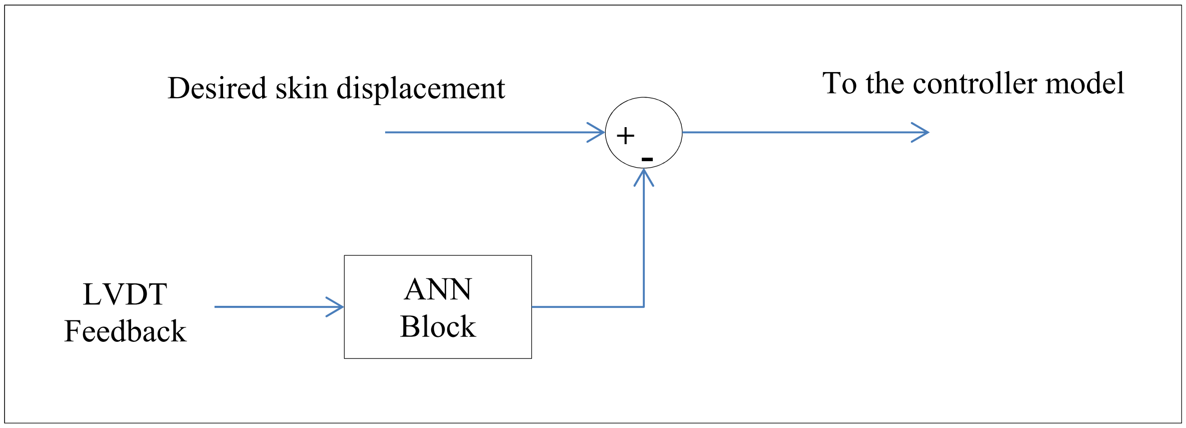


Figure 3.23 ANN system integration

The identified system will be tested under two different conditions, once in the absence of aerodynamic loads outside the wind tunnel and once during actual wind tunnel tests for a number of 15 flight cases to compare aerodynamic responses with those obtained from calibration.

The Figure 3.23 shows the way in which the ANN block was integrated to the control loop. When compared to the previous controller model structure, we see that the set point value is no longer the calibration outcome, but is the actual desired value of the skin. The retrieved value from LVDT is an input to the system identification block and the output value, which is the predicted value, will be subtracted from the desired displacement value, sending the difference to the control model (the one based on feedback from LVDT).

CHAPTER 4

Wind Tunnel Tests and Results

In this chapter, we will provide an overview of the final phase of the project, which involves aerodynamic tests at the NRC-CNRC wind tunnel in Ottawa; only controller experimental results that were obtained during the first two wind tunnel tests, will then be presented.

In total, three test campaigns were carried out. The first campaign took place in April 2015, and was an opportunity to seriously discover the effectiveness and reliability of what had previously been tested and validated on bench test in the absence of aerodynamic loads. It lasted two weeks, with more than half of the time taken by wing system installation and preparation for wind tunnel tests: For example, Kulite sensors installation, the aileron integration, the wing model installation inside the test section, and the calibration procedure are parts of this preparation. A number 38 flight cases in total were then tested.

A second campaign of 97 flight cases took place in June 2015 in order to improve results obtained during the first tests. Finally, the third phase was in November 2015. For these tests, we integrated the morphing aileron, designed and developed by the Italian team from CIRA, Alenia and Naple University. Each of these tests represented a new challenge for the entire LARCASE team, from preparation, through hardware and software integration, to WTT operations.

4.1 Wind Tunnel Tests

4.1.1 NRC-CNRC wind tunnel description

According to wind tunnel geometry classification criteria, there are basically two types of wind tunnels. The first type, referred to as an ‘open return tunnel’ (or open circuit tunnel), has two open sides with airflow from an entry that is open to the atmosphere through the test section. The other type, known as a ‘closed return tunnel’, is characterized by a closed flow circuit, in which the same air recirculates. Usually powered by an axial fan, the stream is conducted from the entry to the test section by four 90-degree corners or turning vanes.

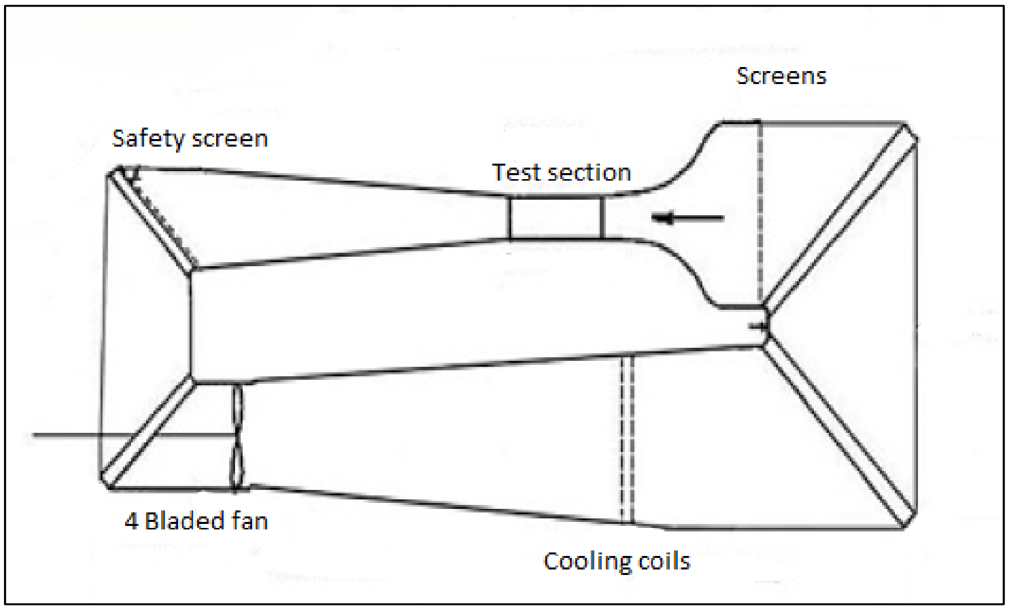


Figure 4.1 Simplified scheme of the wind tunnel at NRC

In our case, aerodynamic tests were performed in the 2×3 meters wind tunnel (Figure 4.1) located in the M2 building of the NRC Institute for Aerospace Research (IAR-NRC). It is classified as a subsonic wind tunnel (or a low-speed wind tunnel), as the ratio of the air speed to the speed of sound is less than 0.3 (Mach number ≤ 0.3). It is also a closed circuit wind tunnel in which air is not pressurized, so that the total pressure (stagnation pressure) is equal to atmospheric pressure. Despite the low-speed range of this wind tunnel, it offers a uniform flow in the test section. The test section is equipped with an aerodynamic balance used to measure forces and torque acting on the tested models. The following table displays some of the technical specifications of the 2 m × 3 m wind tunnel and balance at IAR-NRC.

Table 4.1 Wind tunnel technical specifications
Adapted from National Research Council Canada (2013)

Tunnel geometry	Test section: 1.9 m x 2.7 m x 5.2 m
	Test section area: 5.07 m^2
Tunnel characteristics	Fan power: 1.5 MW
	Maximum speed: 140 m/s
	Speed uniformity: ±0.7%
	Turbulence level: 0.14%
Main balance	Measurement accuracy: ±0.1% to ±0.05% full-scale
	Maximum model weight: 450 kg
	Lift, drag, side force (kN): ±6.7, ±2.3, ±4.4
	Pitch, yaw, roll (kN m): ±2.7, ±2.7, ±2.7

4.1.2 Preparation and performance of the tests

Before starting to perform aerodynamic tests on the wing system, there were some obligatory steps. In fact, the NRC technical team ensures proper installation of the wing prototype inside the test section. The wing is mounted above the turntable structure and should be fixed in such a way so as it would perfectly align the wing chord line with the airflow direction in the working section. This position matches the zero angle of rotation on the turntable so that the wing angle of attack is proportional to the turntable angle.

After installing the mechanical structure, we started establishing necessary connections between the wing model inside the test section of the wind tunnel and the control room. The test section and balance are separated with a pass-through hole for wiring just below the wing root; all hardware wiring (LVDT cables, pressure sensor cables, skin and aileron actuator cables) is driven from the wing to the balance room where it is connected to the real-time system (controller, DAQ, LVDT modules) and motor drives. A simple Ethernet connection from the real-time system to the control room allows controlling the wing system, making it possible for the operator, despite being isolated from the balance room, to establish the communication, check the hardware state, and drive the entire system.

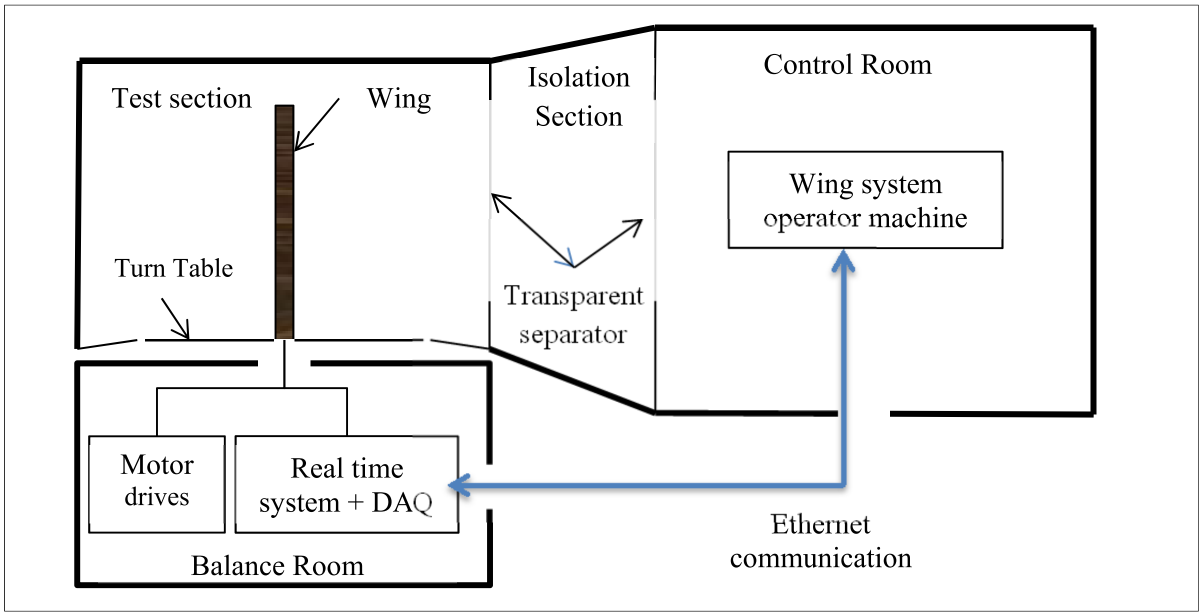


Figure 4.2 Wind tunnel control room, test section and balance room plan

The wind tunnel control center is located inside the control room. The NRC operator is able to control some of the wind tunnel parameters, such as the wind speed, total temperature, and wing angle of attack. The wind tunnel operator is also responsible for taking pressure measurement points, while testing. It is important to note that the NRC and LARCASE teams had two separate data acquisition systems, as pressure measurement technologies were not the same. The pressure distribution on the wing inner surface and on the aileron, are measured through a series of pressure taps installed by the NRC team; pressure data is then acquired by the wind tunnel operator.

Pressure distribution on the upper surface, more exactly on its morphing surface, is measured by double-lined Kulite sensors, since we are especially interested in detecting the transition expected to occur in this zone of the wing upper surface by the use of these sensors. Here, the sensor installation and data acquisition were the responsibility of the LARCASE team.

Once proper installation of the wing, aileron and hardware were verified and performed well, the preliminary preparations took place before the actual tests. The NRC team had to confirm the efficiency and readiness of their wind tunnel control system and data acquisition. Generally, four people were directly involved in test operations: an NRC Test Engineer and Wind Tunnel Operator, and three LARCASE students responsible for wing system control and aileron accelerometer monitoring. In fact, before performing tests on the wing model, our team was obliged by the NRC team to adopt some safety measures; for example, one measure recommended by the NRC team was the 'close monitoring' of aileron flutter during tests in order to detect the occurrence of sudden events. As previously mentioned, one student of our team had to receive accelerometer signals and visualise the acceleration amplitudes in three directions. Data was also logged for further analysis after the wind tunnel tests. Another safety measure was an emergency button that, when activated, cut immediately electrical power to the aileron actuator.

The NRC Test Engineer in collaboration with the LARCASE aerodynamic and control teams must prepare a 'Test Plan' table that lists the execution order of each run. Each row represents a run, and for each run there is some information that should be specified, such as flight case number, wind tunnel speed, wing configuration (morphed or non-morphed), etc. The wind is then turned on and some 'dry runs' carried out. During a dry run, no data is logged, and the NRC performs additional tests using their own equipment, such as:

- Smoke test: This test is used to visualize airflow around the model, and to determine the exact location of the wake rake instrument.

- Turntable test: During this test at low speeds, the operator ensures the turntable is rotating in both directions without any problems.

- Balance limit verification: When the wing is submitted to aerodynamic forces and the balance is used to measure the amplitude of the loads, forces, and moments. This step allows us to validate the analytical aerodynamic results are validated. Then, it is ensured that even in extreme test conditions (high wind speed, significant angle of attack), the loads are lower than the balance loads limits in any direction.

After performing these preliminary verifications, the controller tests can be launched. In terms of testing procedure, communication between different project members is essential, as concentration and synchronization must be maintained. The Test Engineer will announce the 'run' number, which is also visible on a screen. The operator must then adjust the wind speed and the turntable angle for the run. When he is ready, he must confirm, so that the wing system operator can apply the necessary commands for the wing configuration. For each flight case, there are morphed and non-morphed configurations; however, Mach number, angle of attack, and aileron deflection angle remain the same for both configurations at a time. When the wing operator is ready, he must give a signal so that the test engineer can then give the order to both teams (NRC and LARCASE teams) to record data after keeping needed time for flow stabilization. If the two teams have recorded their data, the present run ends and the next run starts.

4.2 Results

4.2.1 Wing shape morphing using lookup table results

Principles of wing shape morphing using lookup tables were explained in the previous chapter (3.3). Some tests were then executed to check the validity of this method, using the first thirty flight cases. The following figures (Figure 4.3 to Figure 4.6) present errors obtained for the four actuators. The Figures below show the error obtained on skin displacement for all actuators. Error is here defined as the absolute value of the difference between the desired skin displacement and the displacement measured with the digital dial indicator, using lookup table control methodology.

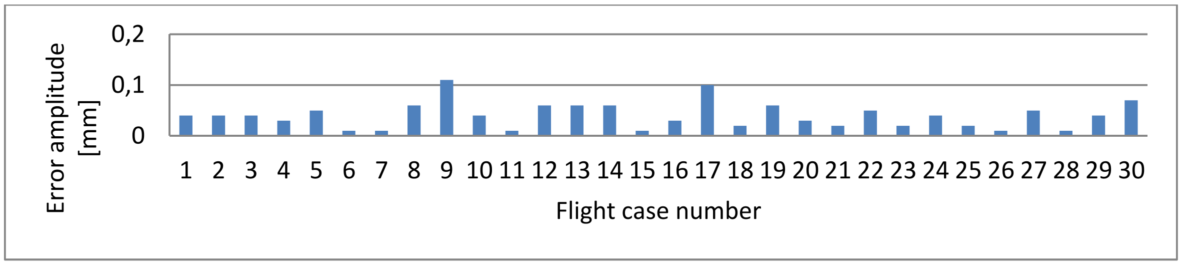


Figure 4.3 Error distribution by flight case for the actuator ‘1’

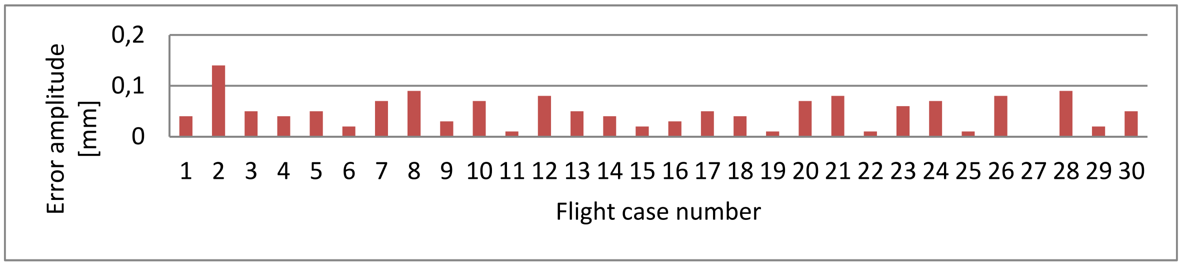


Figure 4.4 Error distribution by flight case for the actuator ‘2’

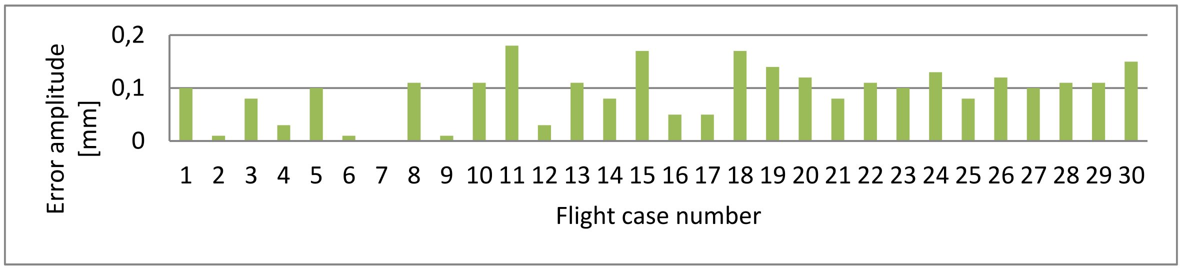


Figure 4.5 Error distribution by flight case for the actuator ‘3’

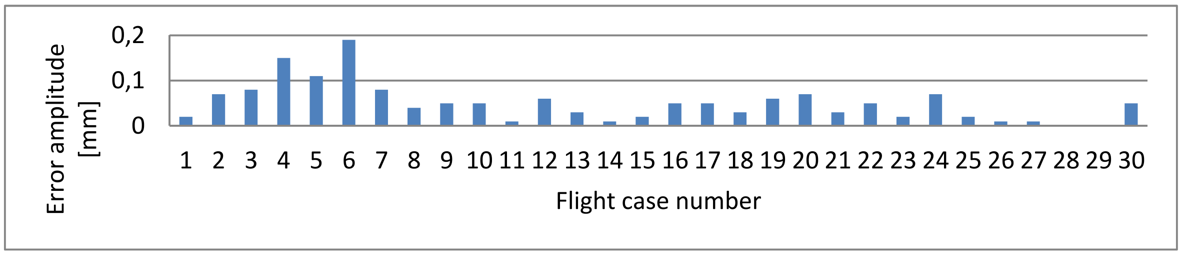


Figure 4.6 Error distribution by flight case for the actuator ‘4’

Error specifications were not respected as requested (error <0.1 mm); consequently, the first actuator performed the best, since the error threshold was respected for all cases but one (flight case number nine in the Figure 4.3). Acceptable results were also obtained with the second and fourth actuators, but it can be clearly observed that displacement error for the third actuator was over the limit of 0.1 mm for more than twelve cases. If error exceeds the requested error limit of 0.1 mm for one actuator out of four actuators, we cannot validate the skin deflection obtained with these actuators displacements, because a wrong value on one actuator will result in a different skin shape different than the desired shape. We will therefore be unable to validate the optimization of the skin shape for each flight case, and instead another shape for the displacement combination is unknown will be observed.

The rationale for this method is the establishment of a continuous domain or a range of displacements where there is a valid relationship between the LVDT indications and the real skin displacements. As we observed, it is difficult to obtain adequate results as the skin displacement above an actuator depends on the positions of other the actuators, and their corresponding displacements.

In addition to being imprecise, this method does not offer much flexibility, as it depends initially on flight case displacement, and will not be valid if we change the flight case displacement values or range; this is the reason why it was not retained as an efficient control method for WTTs.

4.2.2 Wing shape morphing using calibration results

The method retained for use in WTTs was the calibration procedure, since it allowed us to calculate the appropriate set points on LVDTs for all flight cases. This procedure was repeated for all three test campaigns.

4.2.2.1 Preliminary calibration results

The calibration procedure was tested and validated at LARCASE laboratory in the absence of aerodynamic loads. The results obtained for the model calibration and validation that took place before using the wind tunnel are here presented. The principles of the calibration procedure were detailed in the previous chapter (3.5); the validation consists of controlling the wing shape through a closed loop on LVDTs, using calibration values, and digital dial indicators DDIs simply for measuring skin displacement above each actuator.

Figure 4.7 shows errors measured after the calibration procedure. The error calculated here is the difference between the theoretical desired value and the final value measured on DDI during calibration. The maximum error value was measured for the third actuator with 0.05 mm of amplitude; otherwise, errors were ≤ 0.02 mm on all four actuators for 35 flight cases.

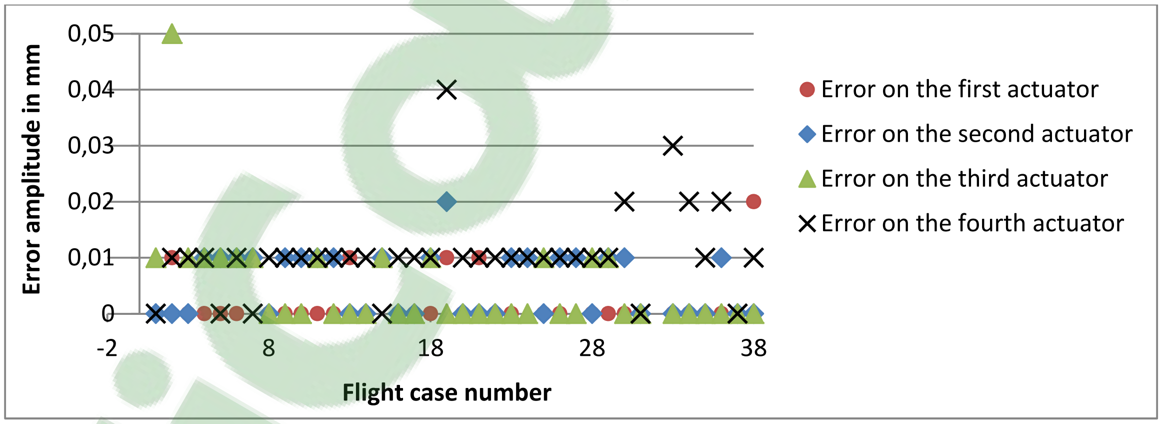


Figure 4.7 Measured skin displacement error for each flight case during calibration

The next step is to implement LVDT values obtained (calibration outcomes) with a closed loop on LVDT feedback. Each flight case is then executed and skin displacement is measured on four actuators points as usual. The next figure illustrates the measured error by referring to the LVDT values. The error given in Figure 4.8 is the difference between LVDT values obtained during calibration and LVDT values obtained during validation (closed loop control position).

clicours.com

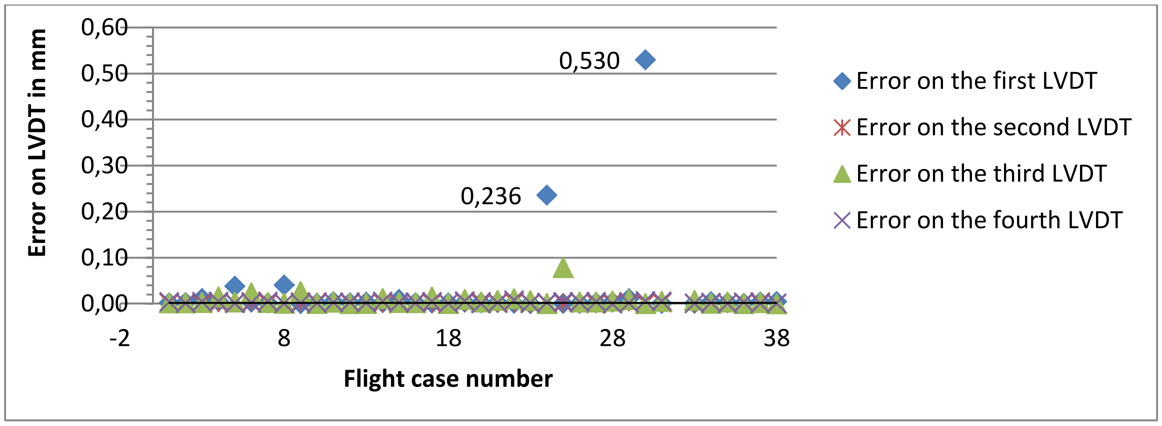


Figure 4.8 Measured error on LVDT during the validation of calibration results

The error value exceeds the amplitude of 0.1 mm for only two flight cases at the same actuator (number 1). Although the results of our first validation attempt are shown, the error disappeared when the experiment was repeated, as the closed position control loop did not produce an error exceeding 0.03 mm during subsequent tests. A mechanical problem on the first actuator may be the reason for the error obtained during our first validation attempt.

The final step is to calculate the cumulative error, which is the difference between the measured skin displacement during calibration and measured skin displacement during validation. This ‘cumulative error’ helps us to quantify our results versus their desired precision. Figure 4.9 shows the error amplitude for all flight cases; as previously observed, the error exceeds the limit of 0.1 mm for only two cases (which are the cases as the ones found during the validation step). Otherwise, the result is acceptable, since all errors are below or equal to the threshold value.

Skin shape optimization using the calibration method has shown high performance in terms of precision and repeatability, and is considered to be the simplest way of reproducing optimized skin shapes. Other advantages of this procedure are its flexibility and fast execution, making it easy to run and adaptable to different situations. Next, we present calibration results achieved during WTTs for two test campaigns.

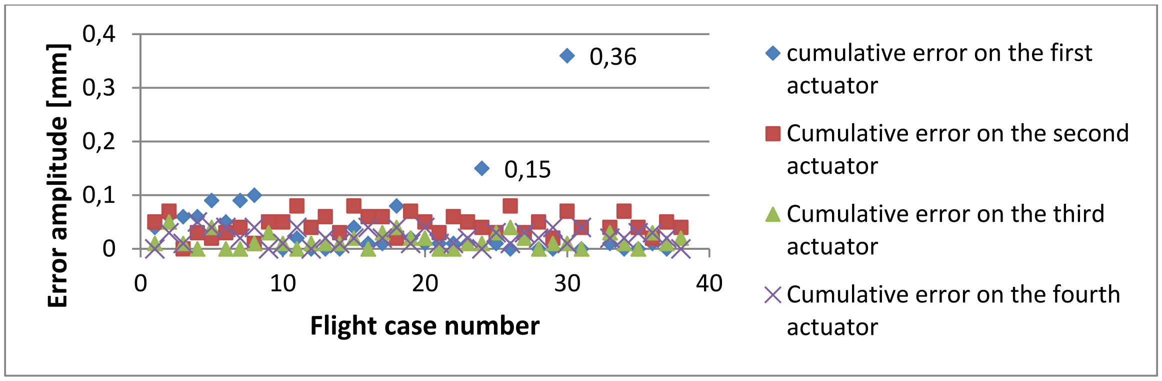


Figure 4.9 Cumulative errors of the skin displacement control using the calibration method

4.2.2.2 Results of WTT calibration

- First test campaign:

For the first campaign, only 38 optimized shapes were planned for testing purposes, and as mentioned in the previous chapter (3.5), calibration took place after installing the wing model inside the wind tunnel at the NRC in order to obtain accurate calibration measures.

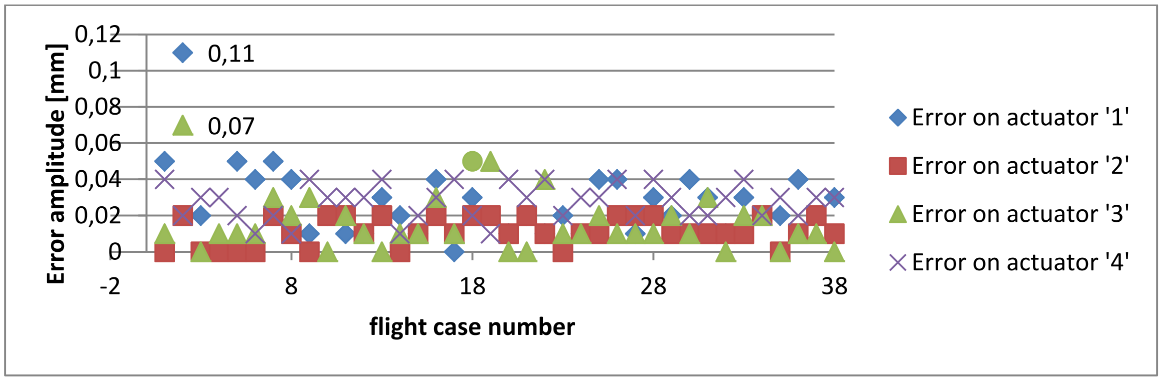


Figure 4.10 Errors obtained during WWT calibration of four actuators

The Figure 4.10 represents the errors of the calibration procedure, consisting of the difference between desired skin displacement values and measured displacement values obtained. Error is calculated separately for each actuator.

As we can see, an error exceeding 0.1 mm of amplitude is obtained for flight case number 2 for actuators ‘1’ and ‘3’. This error is unacceptable within the fixed criteria, so this flight case was repeated separately for more precision. For the rest of flight cases, results were overall acceptable, since errors were smaller or equal to 0.05 mm. As we cannot use digital dial indicators to measure skin displacement with wind, the accuracy of skin shapes compared to desired ones is checked by comparing returned values of LVDTs during aerodynamic tests, with the LVDTs values during calibration (which were used as set points during aerodynamic tests).

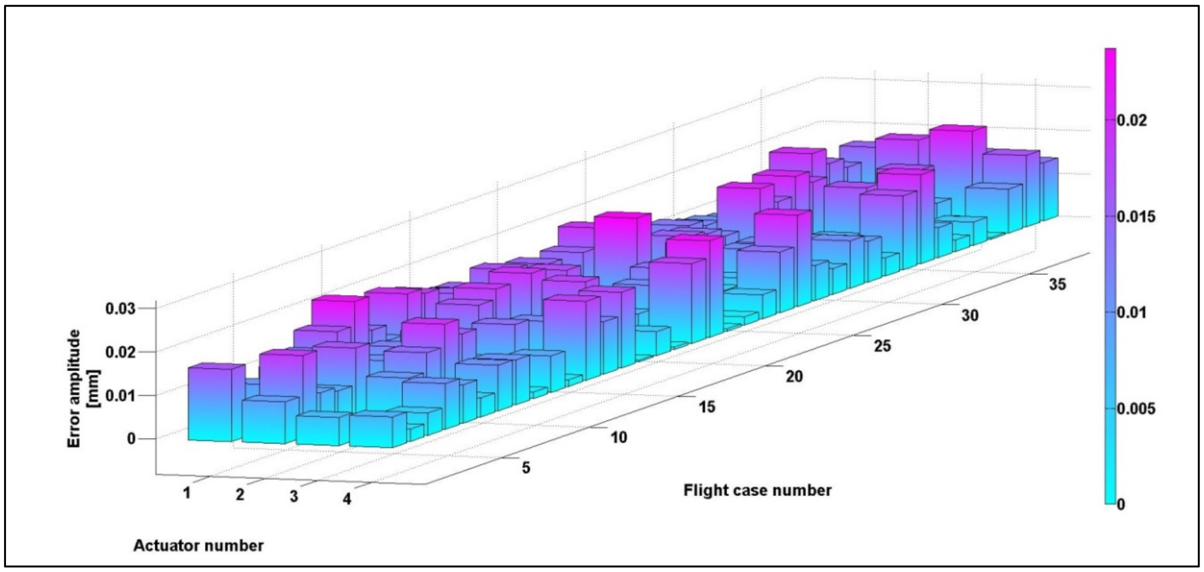


Figure 4.11 Displacement’s error amplitude measured on LVDTs during Wind Tunnel Tests

In the Figure 4.11, displacement error distribution during WTTs in April 2015 is represented accordingly to the actuator number and the flight case. For all 38 flight cases, the error remains below 0.03 mm for all actuators. This error level is reassuring when added to calibration error, since cumulative error would not exceed the level of 0.8 mm in the worst case.

- Second test campaign:

During the second test campaign, the number of flight cases was increased to 97 since several aerodynamic configurations were added to verify the accuracy of the comparison between numerical and experimental results. This pie chart (Figure 4.12) shows the percentage breakdown of errors obtained during the calibration procedure for each actuator.

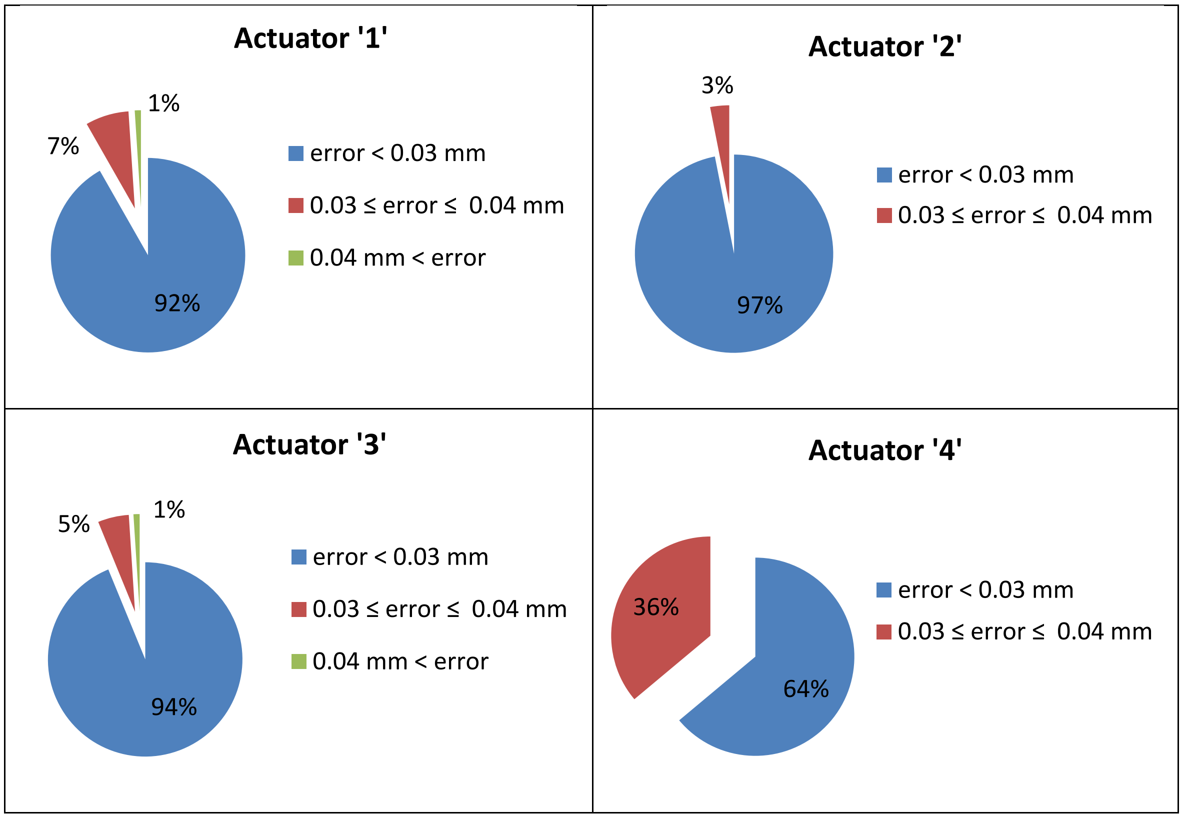


Figure 4.12 Calibration error percentages for each actuator

The calculated absolute error above is the difference between the desired skin displacement value and the displacement value measured during calibration. Error exceeds 0.04 mm of amplitude in only one flight case, number two, where errors were 0.08 mm and 0.06 mm respectively, on the first and third actuators. The flight case was recalibrated in order to obtain acceptable error amplitude (generally a value lower than 0.05 mm) and an error of 0.03 mm was obtained. For the rest of the flight cases, the error did not exceed 0.04 mm of amplitude for all actuators as seen on Figure 4.12.

4.2.3 Overview of aerodynamic results

4.2.3.1 Overview of the method adopted for transition detection

In order to detect the flow transition, two methods were basically used: The first method was based on high-sampled pressure measurements that were used by the LARCASE team and the second is a thermal method used by the NRC team. The validation step consists of comparing the transition location obtained using these two experimental methods with CFD numerically-predicted transition points.

The method adopted by the LARCASE team consists of collecting pressure data (waveform logging) with a very high sampling rate (20 kHz), then, of detecting the exact location of the transition through frequency analysis.

- Fast Fourier Transform (FFT) Waveform Analysis

> A fast Fourier transform (FFT) is an algorithm that computes the discrete Fourier transform (DFT) of a sequence, or its inverse. Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa. Taken from (WIKIPEDIA (2015a))

The Figure 4.13 shows the FFT decomposition in the frequency domain of 32 waveforms for a morphed upper surface wing configuration. According to the power spectrum amplitude, we can see that there are mainly two levels: the first level is situated between -120 and -110 dB of amplitude; while the second level is situated between -100 and -90 dB. These two levels correspond, respectively, to the power spectrum of each group of Kulite sensors before and after the flow transition. As the flow becomes more turbulent after the transition, the power spectrum should have greater amplitude than before the transition (when the flow is almost laminar). The black arrow points to the channel '11', where the transition starts and then we observe that amplitudes increases on the next channels.

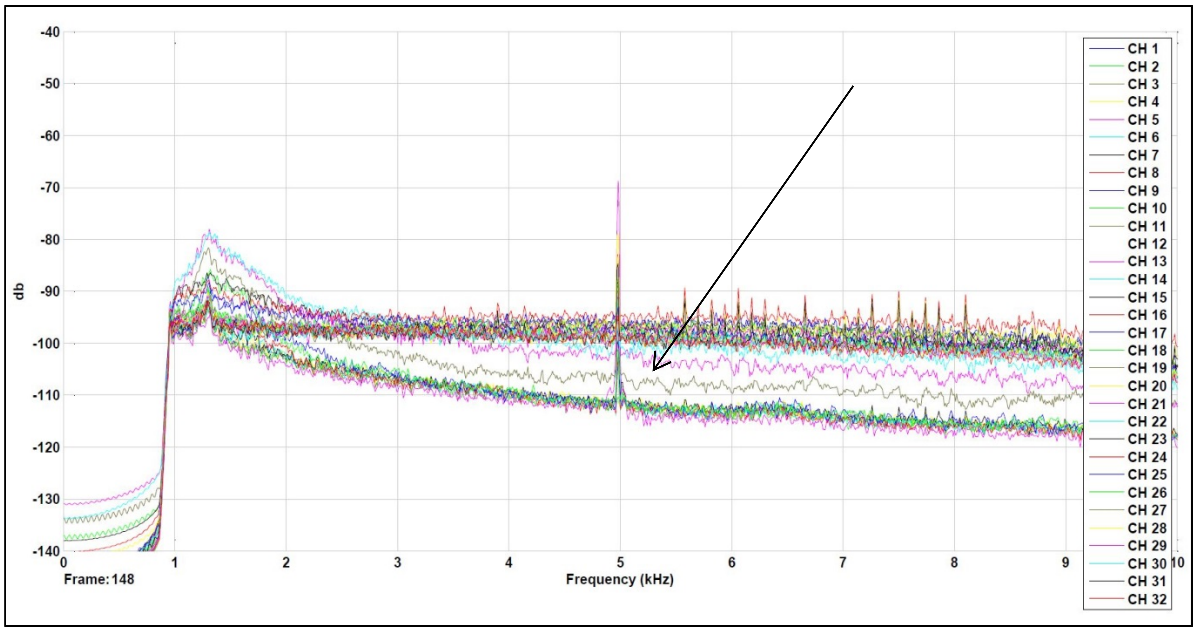


Figure 4.13 Power spectrum of Kulite sensors for a morphed configuration

For more accuracy, standard deviation of the pressure data is also calculated as following:

- Standard deviation (STD)

$$\sigma_{\Delta P} = \sqrt{\frac{1}{N}\sum_{N}^{1}(\Delta P_i - \Delta P)^2} \tag{4.1}$$

Taken from WIKIPEDIA (2015c)

This is a measure of the distribution of pressure values obtained by calculating the square root of the variance which is defined as the average of the squared pressure differences from the mean.

Standard deviation is calculated for the same data buffer as for the FFT power spectrum. The peak on the curve shows the channel 13, that has the greatest value of STD. This fact confirms the hypothesis of the FFT curve analysis; indeed, one could say the transition is clearly established at the sensor number 13 which is localized at 44% of the chord.

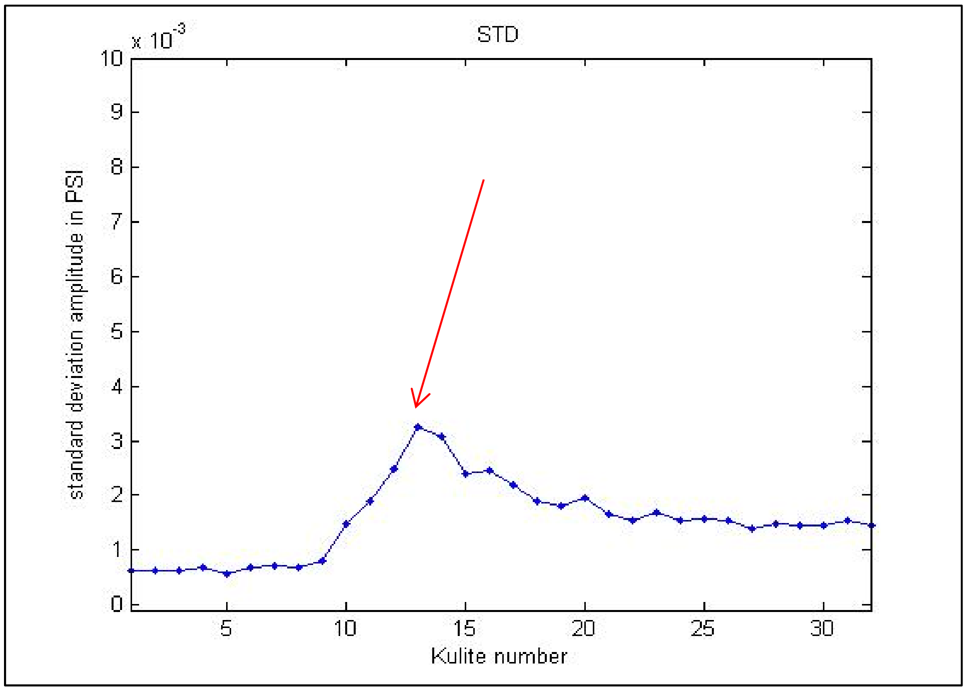


Figure 4.14 Standard Deviation (STD) of pressure data for all channels

Another method for transition detection adopted by the NRC team is used. This method consists in performing some infrared measurements using a Thermal Camera. The flexible skin is painted with a high emissivity black coat, then is heated using light projectors. IR-captured images will display two different color scales according to the temperature, since thermal conductivity of the laminar flow zone is different from thermal conductivity of the turbulent flow zone.

The Figure 4.15 is an example of IR pictures taken by the NRC. We observe a color difference between the laminar flow region (light blue, upstream from the continuous line) and the turbulent flow region (yellowish, downstream from the continuous line). We should also mention that the transition is always determined with some uncertainty due to the multitude of physical and experimental factors that affects the overall method used for transition detection.

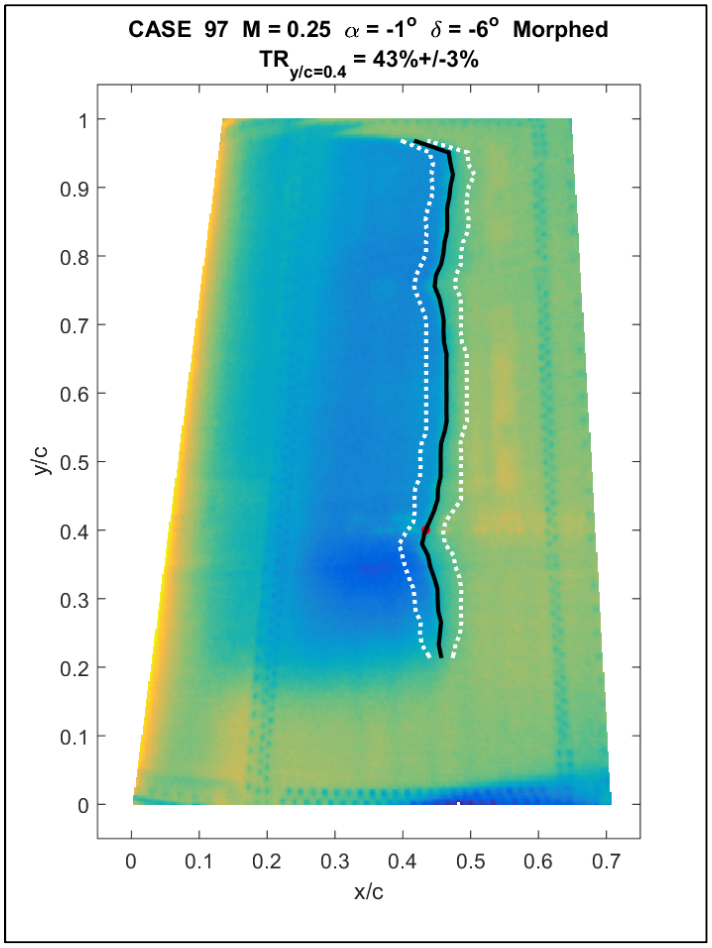
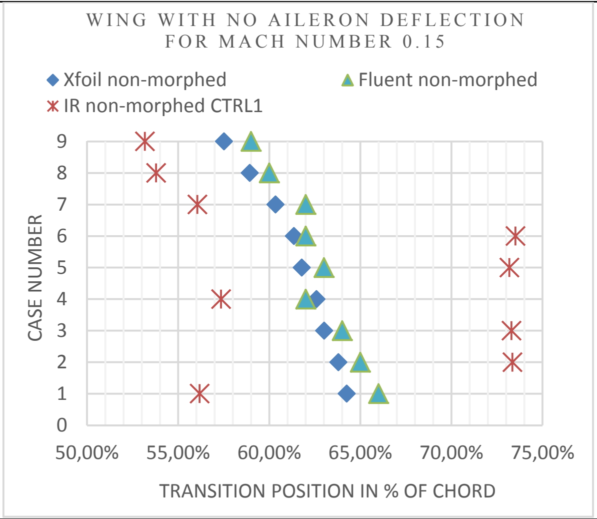
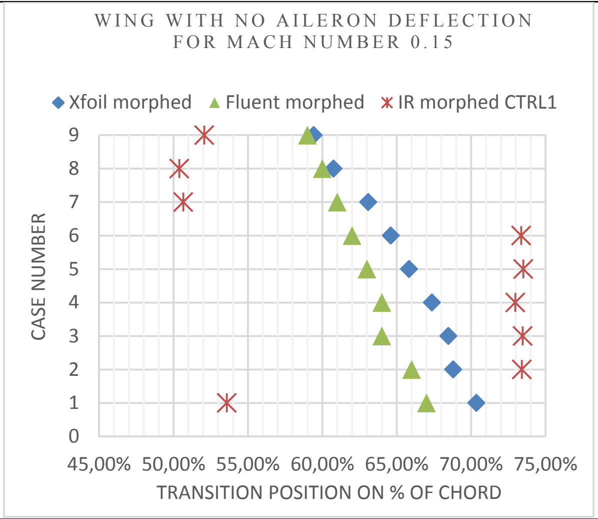
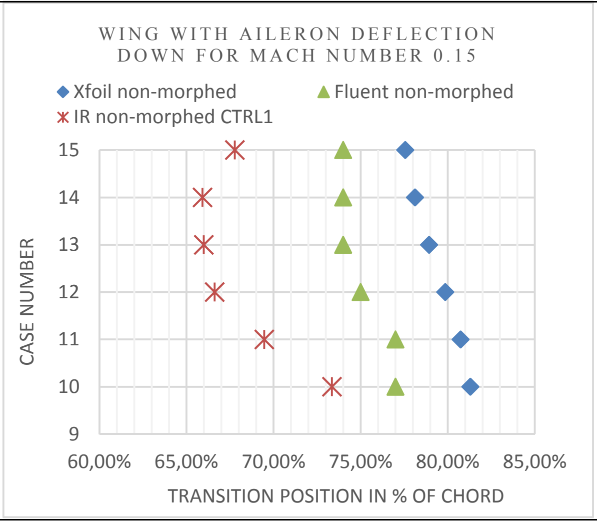
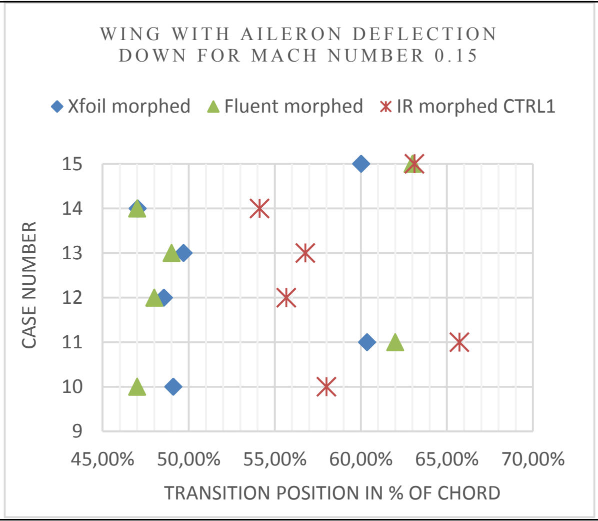


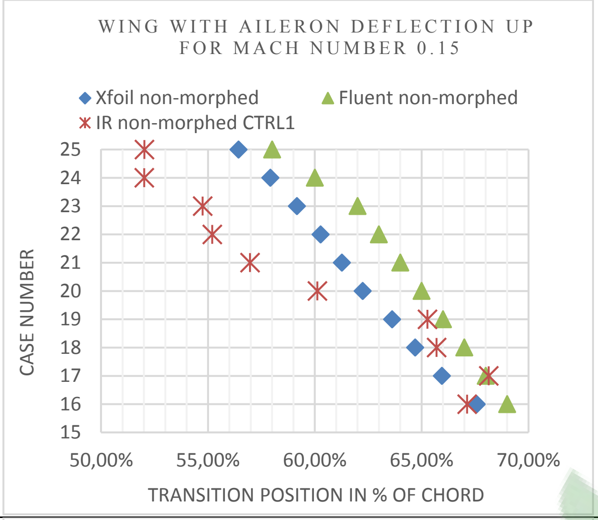
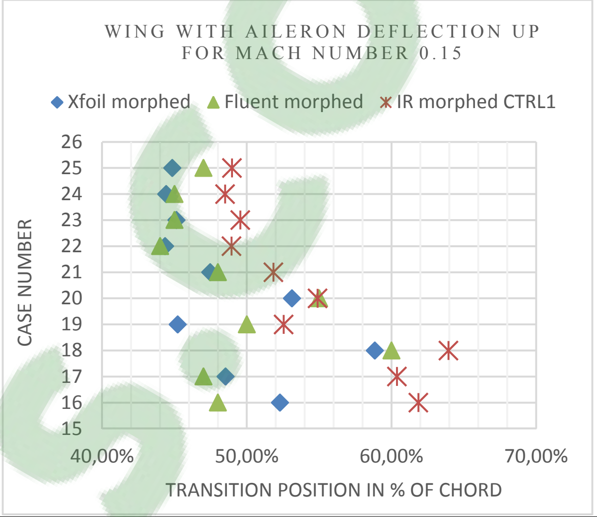
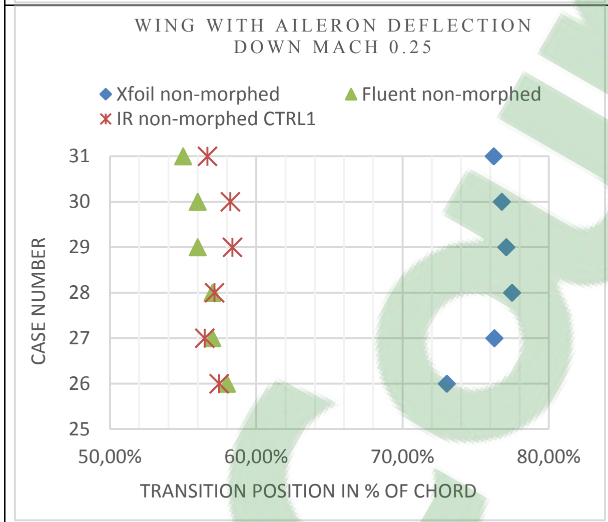
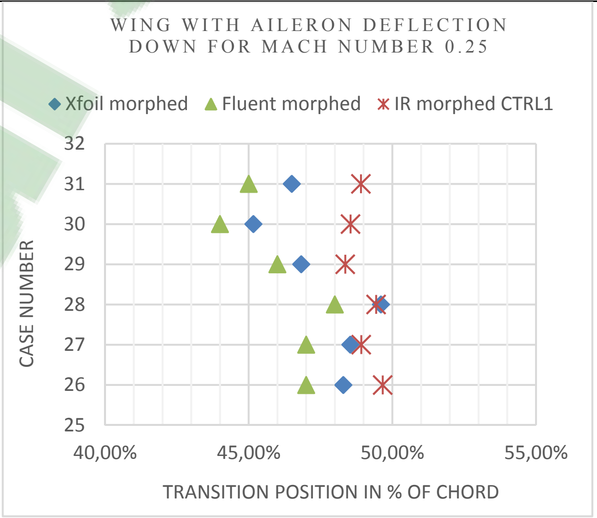
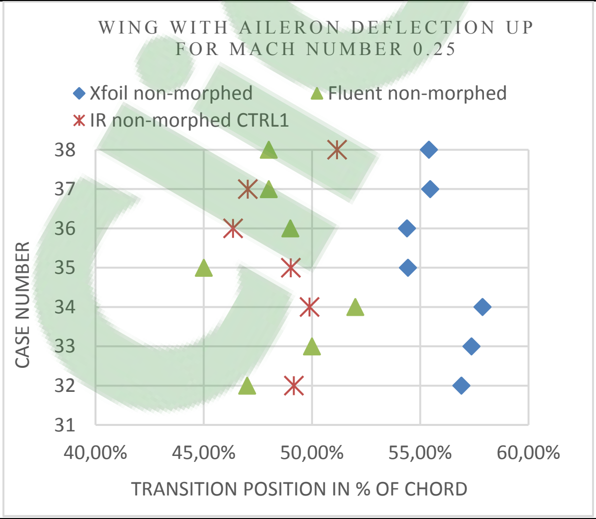
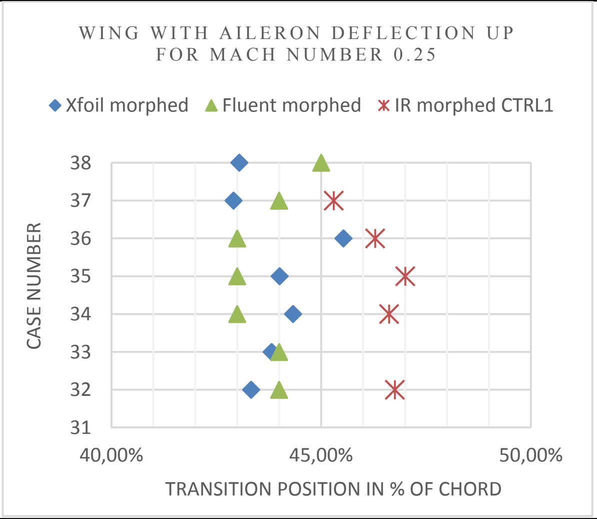
Figure 4.15 Infrared picture of the wing upper surface with a morphed configuration

4.2.3.2 Global overview of aerodynamic results

The main objective behind all the control work is purely aerodynamic optimization. For this reason, the comparisons between experimental and numerical results are here presented.

Table 4.2 Comparison between IR transition detection and numerical transition prediction during the first WTT campaign

Non morphed configuration	Morphed configuration
	
	

Non morphed configuration	Morphed configuration
	
	
	

- IR measurements versus numerical analysis for transition prediction

The Table 4.2 shows a comparison between transition location using the NRC's IR method and two numerical aerodynamic analysis methods used during the first test campaign. The two methods of analysis are identified as "Xfoil" and "Fluent" in the graph legends. "Xfoil" in fact refers to '2'-dimensional analysis software as it only takes into account airfoil dimensions and geometry. "Fluent" refers to a '3'-dimensional analysis method, as it takes into account 3-dimensional wing geometry.

For the first test campaign, optimization of skin displacement is performed using a 2D analysis method (Xfoil) on a theoretical wing airfoil. These optimized skin shapes are then used to reconstruct geometry with the 3D analysis method, and to perform CFD analysis transition prediction.

It is therefore extremely odd that there is a significant difference (seen in the last two rows of the Table 4.2) between the transition locations predicted by each numerical analysis method. The difference between experimentally-detected transition and numerically-predicted transition varied between 0.5% and 10% of the wing chord. An error of less than 10% of the chord can be explained by uncertainties in CFD analysis, differences between theoretical and experimental conditions, and even uncertainties related to IR experimental results.

After the first WTT campaign, a scan wing operation was performed in order to validate its skin shape deflection. As expected, displacements were right measured above each actuation point (we have already shown the validation results using DDIs), but the problem appeared in the zone between two actuators on the same chord. In fact, the skin behaved strangely and did not respond correctly to actuator displacement, and some zones moved down where they were supposed to move up. We concluded, without going into too much detail, that the interactions between skin and actuators were not enough optimized.

A technical problem has occurred and it was related to the data acquisition system of the Kulite sensors, and we were unable to log pressure data for cases with Mach number = 0.25, which was the highest Mach number considered among the 38 cases. As a result, we present a comparison with Kulite transition for only 25 test cases.

- Transition position for non-morphed configuration:

  For the non-morphed configuration, a comparison between numerical transition predictions and experimental methods is presented in the Figure 4.16. Error between IR-detected transition and Fluent-predicted transition occurs at amplitude below 5% of chord length for 28% of the flight cases while 64% of the cases show errors between 5% and 10% of chord length and only two cases show higher than10% of chord errors (see Tableau-A X-1 for detailed results).

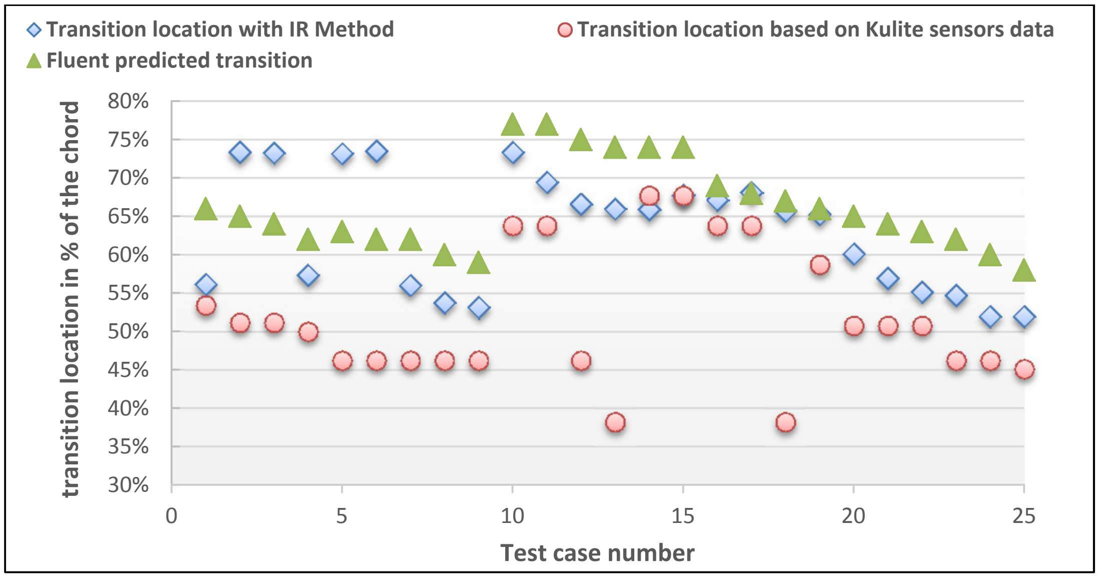


Figure 4.16 Comparison of the transition with different detection methods for the non-morphed wing

Seen from a different perspective, a comparison between IR-detected transition and Kulite-detected transition shows an absolute error below 5% of chord length for 24% of the test cases. 48% of the cases show between 5% and 10% difference in chord length and 28% of the cases presented an error above 10% of chord length.

- Transition position for morphed configuration

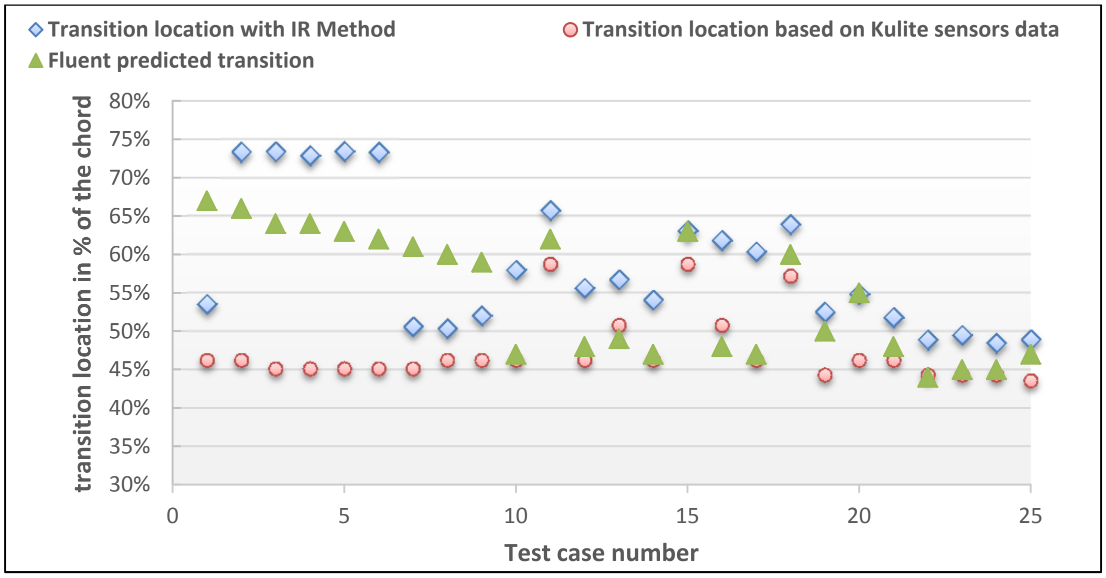


Figure 4.17 Comparison of the transition with different detection methods for the morphed wing

The same comparison was made for the morphed configuration as for the non-morphed configuration. Results shown in Figure 4.17 are presented in detail in Tableau-A X-2. Error was found to be below 5% difference between the predicted transition using 3D analyses and detected transition using the IR method for 40% of the cases among the first 25 cases. For 32% of the cases, absolute differences between the 'IR method' versus 'Fluent method' results were between 5% and 10% of the chord. Regarding the differences between the Kulite-detected transition and IR method results, only 16% of the cases presented errors below 5% of the chord, while 48% of the cases out of 25 have shown error in the range of 5% to 10% of the chord length.

Note that for this first WTT, the flow transition using the 'Kulite' or the 'IR method') was considered to be a single point along the chord, so that measurement error uncertainty was not considered. After this first test, IR and Kulite transition detection methodologies were improved and the transition was perceived as an entire region rather than as a specific point. During the second WTT campaign, scanned shapes were used for CFD analysis to improve transition prediction analyses. The number of flight cases increased from 38 test cases to 97 in order to produce additional test conditions and better understand various project-related issues.

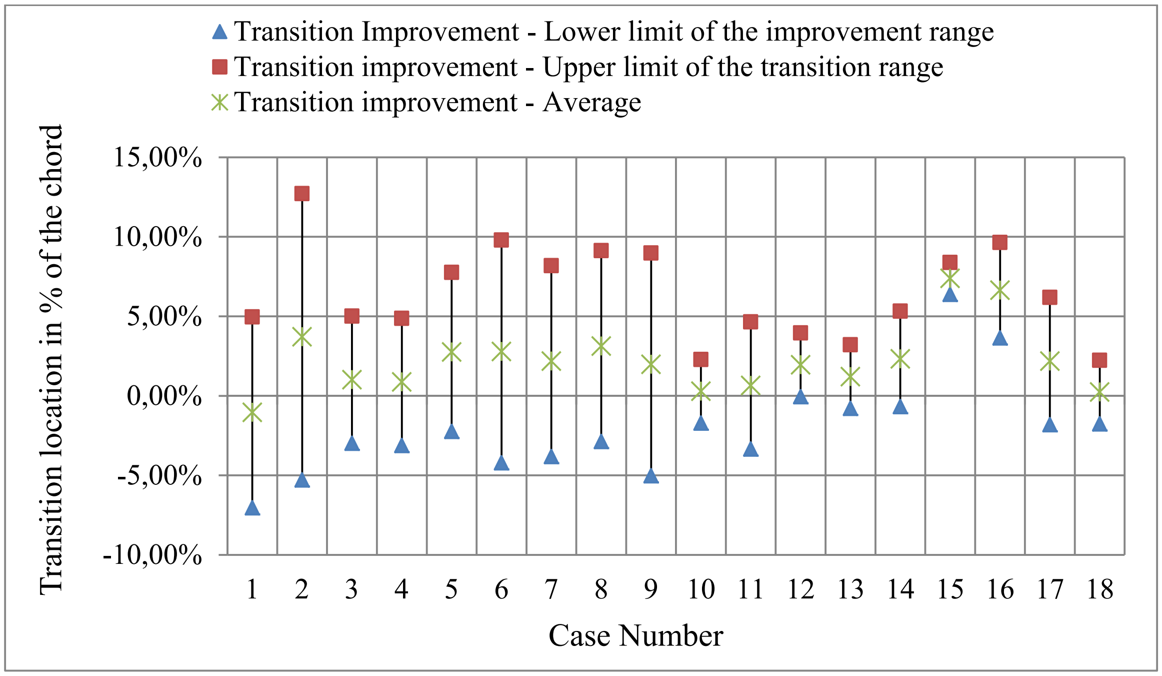


Figure 4.18 Experimental results of the 18 cases improved for transition delay

Here, we present conclusions regarding the final results only, rather than discussing all the cases in details:

- Since methodologies for transition detection were improved, the first 38 cases were repeated with the aim to compare the first WWT with the second WTT cases results, as follows:

  -Un-morphed configuration results: The second WTT IR measurements recorded a more laminar flow than during the first series of tests, with an average error between IR transition measurements of 5.22% of the chord.

-Morphed configuration results: The average error between IR transitions from the two WT tests is 4.86% of the chord length. Also, with the exception of case 12, all the results indicate that the second IR measurements registered a more laminar flow than the first series of tests.

- Almost 30% of the cases were optimized for transition delay towards the trailing edge.
- All other 70% of cases were optimized for transition advancement towards the leading edge.
- Out of 30 cases optimized for transition delay, 18 cases were confirmed experimentally through infrared and Kulite methods for transition detection (Figure 4.18).

CONCLUSION

In this thesis, we have presented our contribution in the realization of the MDO 505 project, which consists of the integration and control of the morphing wing system. Having the opportunity to work on such a multidisciplinary project has been a very rewarding experience on many levels.

The literature review was the key to establish a good understanding of the main issues of this project. Furthermore, the previous work realized on the CRIQA 7.1 project made a major contribution in determining where we should start, and helped to know how to deal with the challenges.

The modular hardware architecture of NI systems has made our task easier, as it afforded a simple way to connect several wing peripherals with different I/O signals to the control unit. It was possible to process the data from 32 pressure channels sampled at 20 KHz and four analog LVDT sensor signals, as well as to manage a CAN-based network with four field devices (Epos2 drives). Useful features such as Ethernet ports helped to implement the communication with the aileron drive, so that the entire system could be monitored remotely from the control room during wind tunnel tests.

Another main part of our work was the integration of communication tools. In order to modify the skin shape of the wing, CANopen communication was established to control the motors positions based on the 'LVDT sensors' feedback. The differential bus signaling of the 'CAN' field network makes it robust regarding EMI noise inside a wind tunnel. The 'CANopen' application layer allowed high configuration flexibility, offering the ability to program devices by choosing a specific performing mode, and by mapping the drive parameters for real time data broadcasting, that was the basis of the subsequent position closed loop. The Modbus TCP protocol was easy to use as it did not require a special interface to physically connect to the PXI-e controller. Using motion tasks with the aileron position control has greatly simplified the protocol implementation, which was based on elementary read/write operations on Modbus registers.

After setting up the communication mechanisms, it was unrealistic to maintain the position that a simple open loop command would be sufficient to obtain accurate skin displacement control. After the starting of the skin shape control operations, several issues began to appear. The first (and quite major) issue concerned that the skin displacement was not matching the desired input value. Several tests led us to better understand the actuator response, which was nonlinear and not repeatable (uncertain reference point); we concluded that some mechanical issues (play in the gearbox, linkage between components) were the main causes. Problems such as 'dead zone nonlinearity' and 'uncertain reference point' were solved by using a closed position control loop based on LVDT sensor feedback. The PID controller was tuned using the 'Hardware in Loop' (HIL) approach; the 'inline gains' were tuned until the optimal actuator response was obtained. The use of the closed loop strategy made possible the indirect control of the skin displacement with a level of efficiency and reliability.

Using look-up tables to determine the relationship between skin displacements and their corresponding LVDT indications was not practical and had, also, poor precision. The calibration procedure was further adopted to determine the LVDT values required to match desired skin displacements. Based on data sets of the desired skin displacement values for each flight case it was possible to retrieve the LVDT set point values for actuator displacements. Automating the procedure made it easy to run and much less time consuming, especially in the wind tunnel environment.

The efficiency and the robustness of the developed communication and control methods were investigated during the bench tests firstly in the absence of aerodynamic loads. The wind tunnel tests proved that we could operate the wing control system directly, and closely monitor the tests' progress during two campaigns. We achieved our goal regarding control and integration, as we did not have any kind of communication failure or control instability.

The control results and more precisely, the static errors on each actuator for the position control loop were also given in details in the thesis. Globally, errors were found to be within the desired limits. In certain cases, errors exceeded the limit (only during calibration), but it was easy to repeat them given the flexibility of the calibration procedure.

Clicours.COM

RECOMMENDATIONS

After the completion of this work and the familiarization with the software and hardware environment, some issues are proposed, that could be further studied and improved:

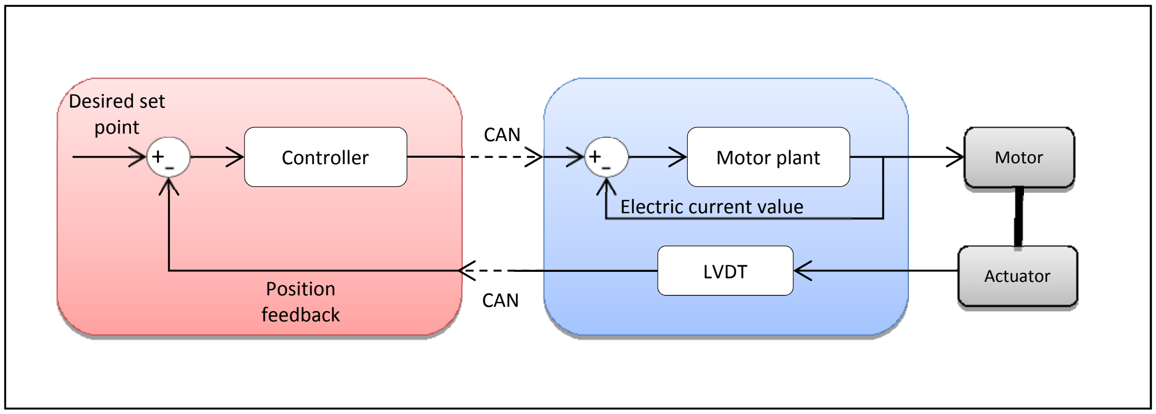
- Because the short deadlines we had in the project, it was not possible to develop a complete numerical model of the wing. Ideally, this model would take into account the actuator's mechanism, motors, and even the composite skin behavior. The main goal of this wing modelling is to (at least) estimate the efforts to be applied on each actuator for different morphing configurations. This estimate would be very useful to allow the position loop controller to be manually tuned.

- Building on our concept of having a complete numerical model of the wing, one could exploit the internal control structure of EPOS2 drives (basically a PID controller) and try to tune the gains of this internal structure using the classical approach for control design (including stability analysis). In fact, since we have established the tools required to program an EPSO2 drive using the CANopen protocol, it would be possible to change the internal control gains via SDOs when configuring the drive. The obtained gains could be compared with the default EPOS2 gains to be sure that they are in the same range.

- Another interesting idea is to change the control structure completely. In fact, since EPOS2 drives support several working modes, they could be configured to perform in 'current mode', and thus our set point to the drive would be a current (electric current) value. The position loop could be completely developed and implemented using a control model programed in Matlab Simulink or LabVIEW. The advantage of using such a concept is that it offers the possibility of implementing complex control algorithms rather than only using the PID method.

clicours.com

- The calibration procedure could be analyzed to determine a large error was obtained for one or two cases.



Control loop with current reference value fed to drive

ANNEX I

Morphing mechanism

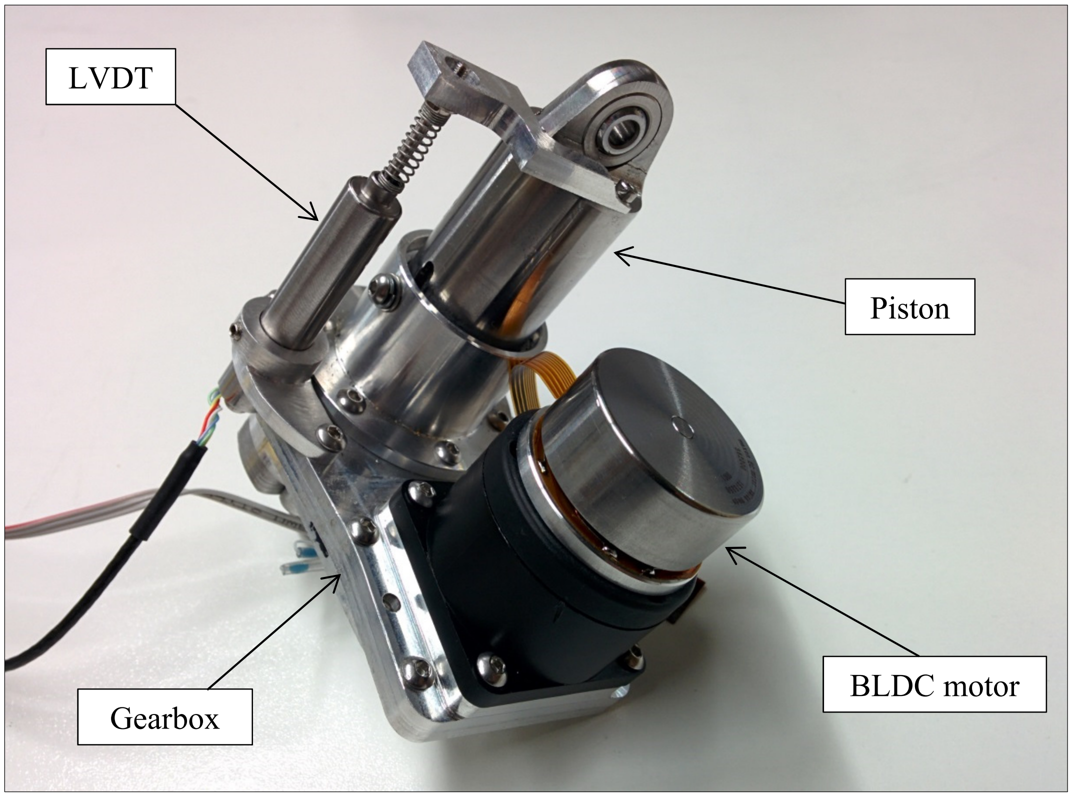


Figure-A I-1 Actuator with LVDT mechanism

ANNEX II

NMT slave state

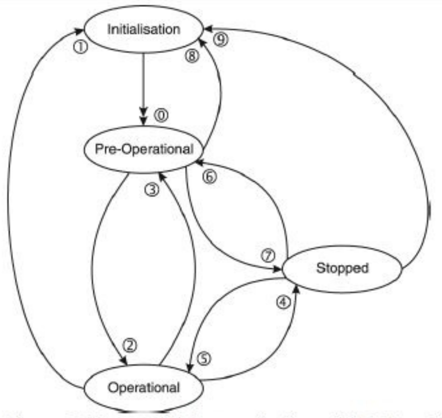


Figure 6-27 CAN Communication – NMT Slave State Diagram

CANopen Network Management provides the following five services, which can be distinguished by the Command Specifier (CS).

Remarks:

*1) Command may be sent with Network Management (NMT) Protocol.
*2) This Transition is generated automatically by the EPOS2 after initialization is completed. After initialization a Boot-Up message is send.
*3) Remote flag Bit 9 of the Statusword.

Service *1	**Transition**	**NMT State after Command**	**Remote** *3	**Functionality**
–*2	0	Pre-Operational	FALSE	Communication: – Service Data Objects (SDO) Protocol – Emergency Objects – Network Management (NMT) Protocol
Enter Pre-Operational	3, 6	Pre-Operational	FALSE	
Reset Communication	1, 8, 9	Initialization (Pre-Operational)	FALSE	Calculates SDO COB-IDs. Setup Dynamic PDO-Mapping and calculates PDO COB-IDs. Communication: – While initialization is active, no communication is supported. – Upon completion, a boot-up message will be sent to the CAN Bus.
Reset Node	1, 8, 9	Initialization (Pre-Operational)	FALSE	Generates a general reset of EPOS2 software having same effect as turning off and on the supply voltage. Not saved parameters will be overwritten with values saved to the EEPROM using «Save all Parameters».
Start Remote Node	2, 5	Operational	TRUE	Communication: – Service Data Objects (SDO) Protocol – Process Data Objects (PDO) Protocol – Emergency Objects – Network Management (NMT) Protocol
Stop Remote Node	4, 7	Stopped	FALSE	Communication: – Network Management (NMT) Protocol – Layer setting services (LSS) – Lifeguarding (Heartbeating)

Figure-A II-1 NMT slave state diagram
Taken from Maxon motor control (2013b)

ANNEX III

Drive states and transitions

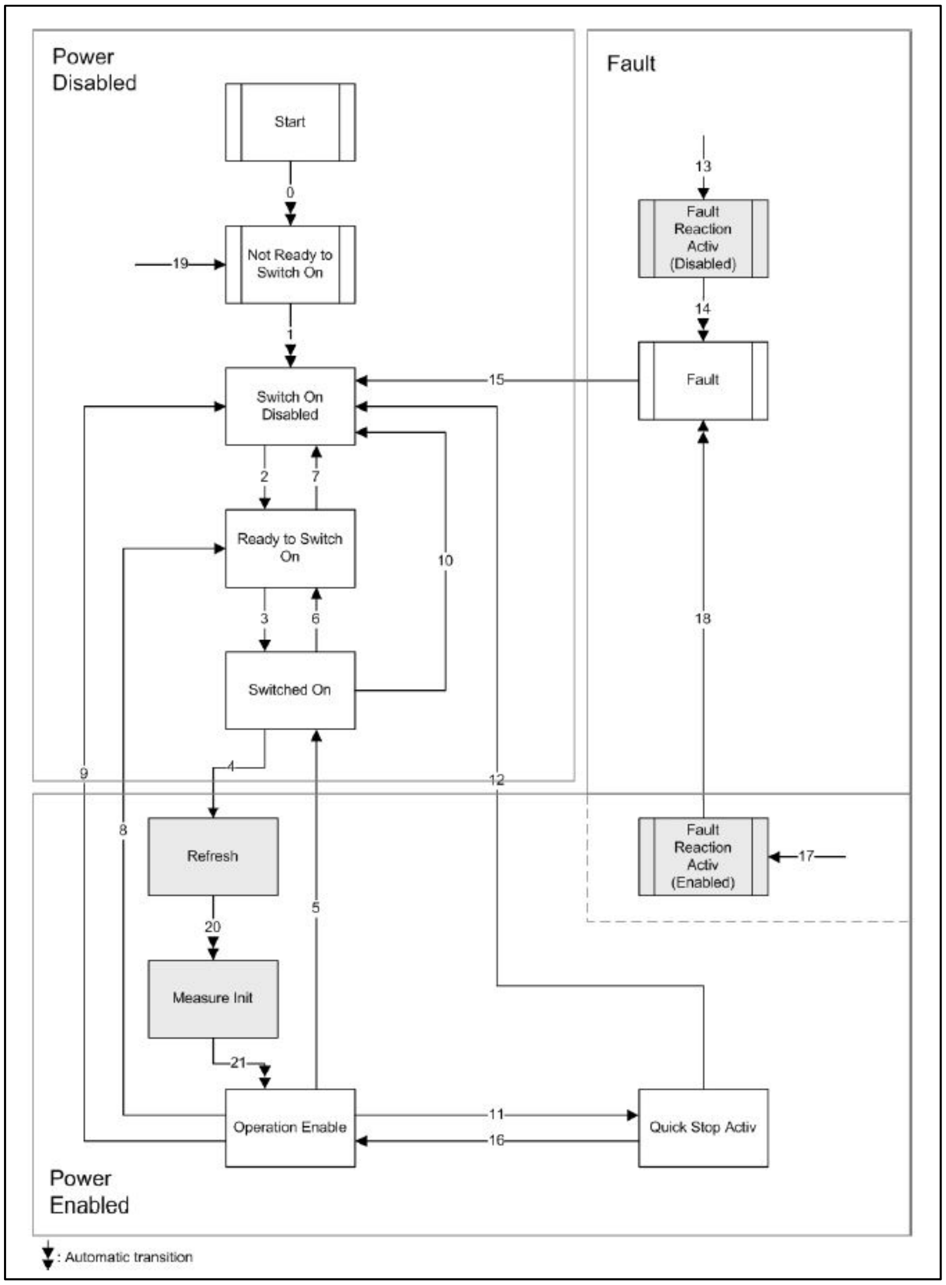
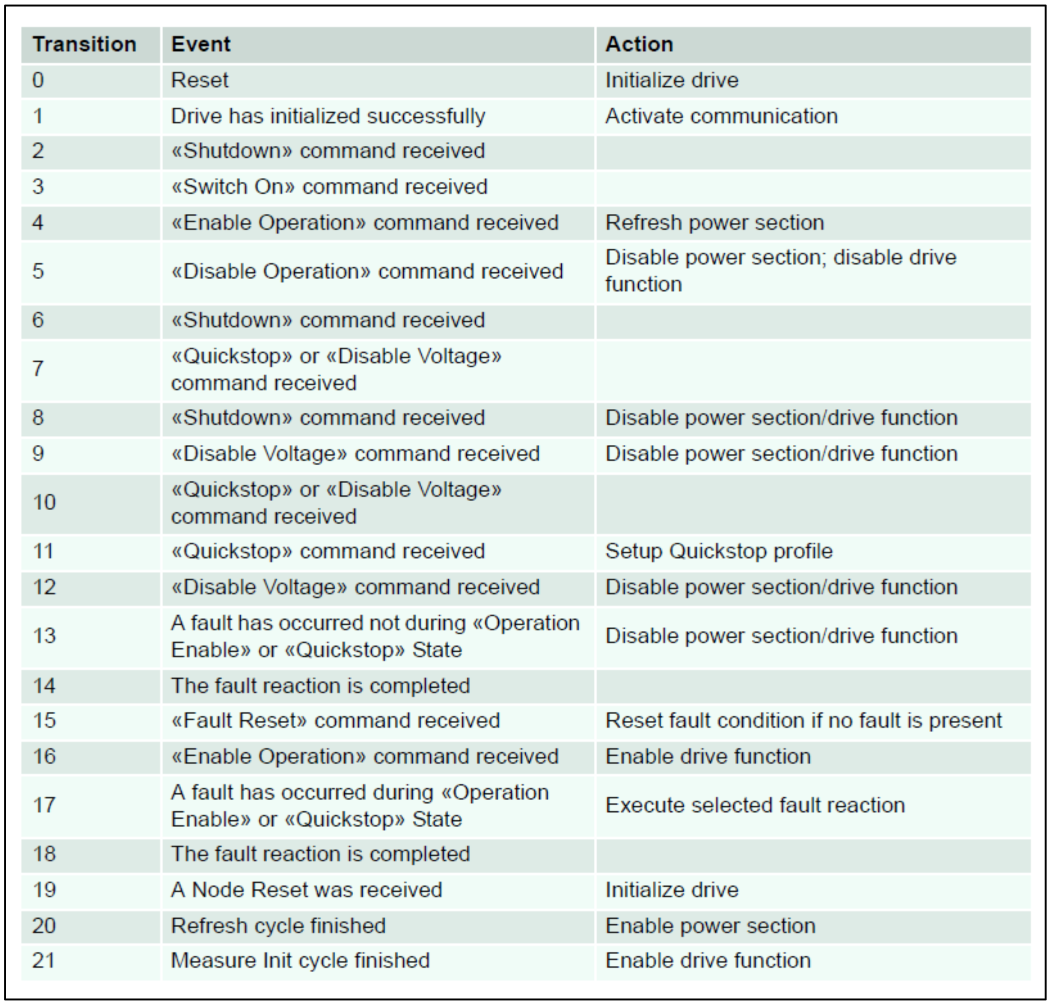
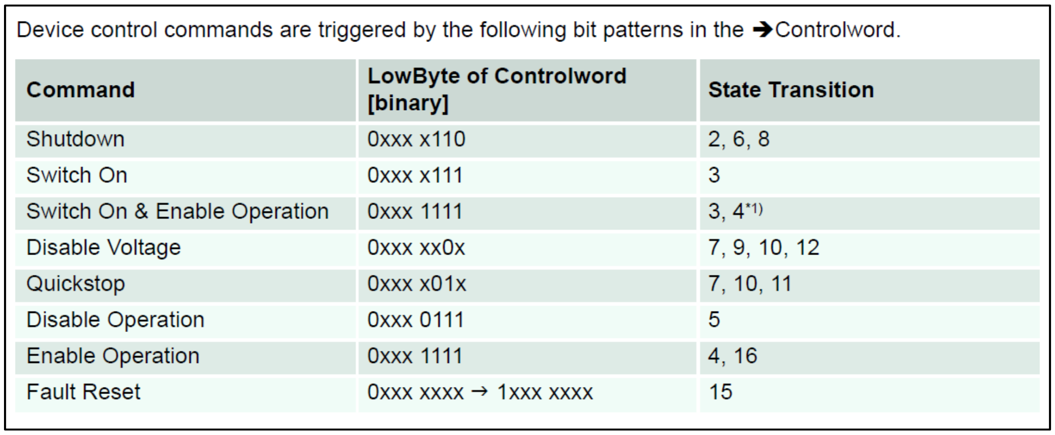


Figure-A III-1 Drive sates and transition diagram
Taken from Maxon motor control (2014c)



Transition	Event	Action
0	Reset	Initialize drive
1	Drive has initialized successfully	Activate communication
2	«Shutdown» command received	
3	«Switch On» command received	
4	«Enable Operation» command received	Refresh power section
5	«Disable Operation» command received	Disable power section; disable drive function
6	«Shutdown» command received	
7	«Quickstop» or «Disable Voltage» command received	
8	«Shutdown» command received	Disable power section/drive function
9	«Disable Voltage» command received	Disable power section/drive function
10	«Quickstop» or «Disable Voltage» command received	
11	«Quickstop» command received	Setup Quickstop profile
12	«Disable Voltage» command received	Disable power section/drive function
13	A fault has occurred not during «Operation Enable» or «Quickstop» State	Disable power section/drive function
14	The fault reaction is completed	
15	«Fault Reset» command received	Reset fault condition if no fault is present
16	«Enable Operation» command received	Enable drive function
17	A fault has occurred during «Operation Enable» or «Quickstop» State	Execute selected fault reaction
18	The fault reaction is completed	
19	A Node Reset was received	Initialize drive
20	Refresh cycle finished	Enable power section
21	Measure Init cycle finished	Enable drive function

Figure-A III-2 State Transitions details
Taken from Maxon motor control (2014c)



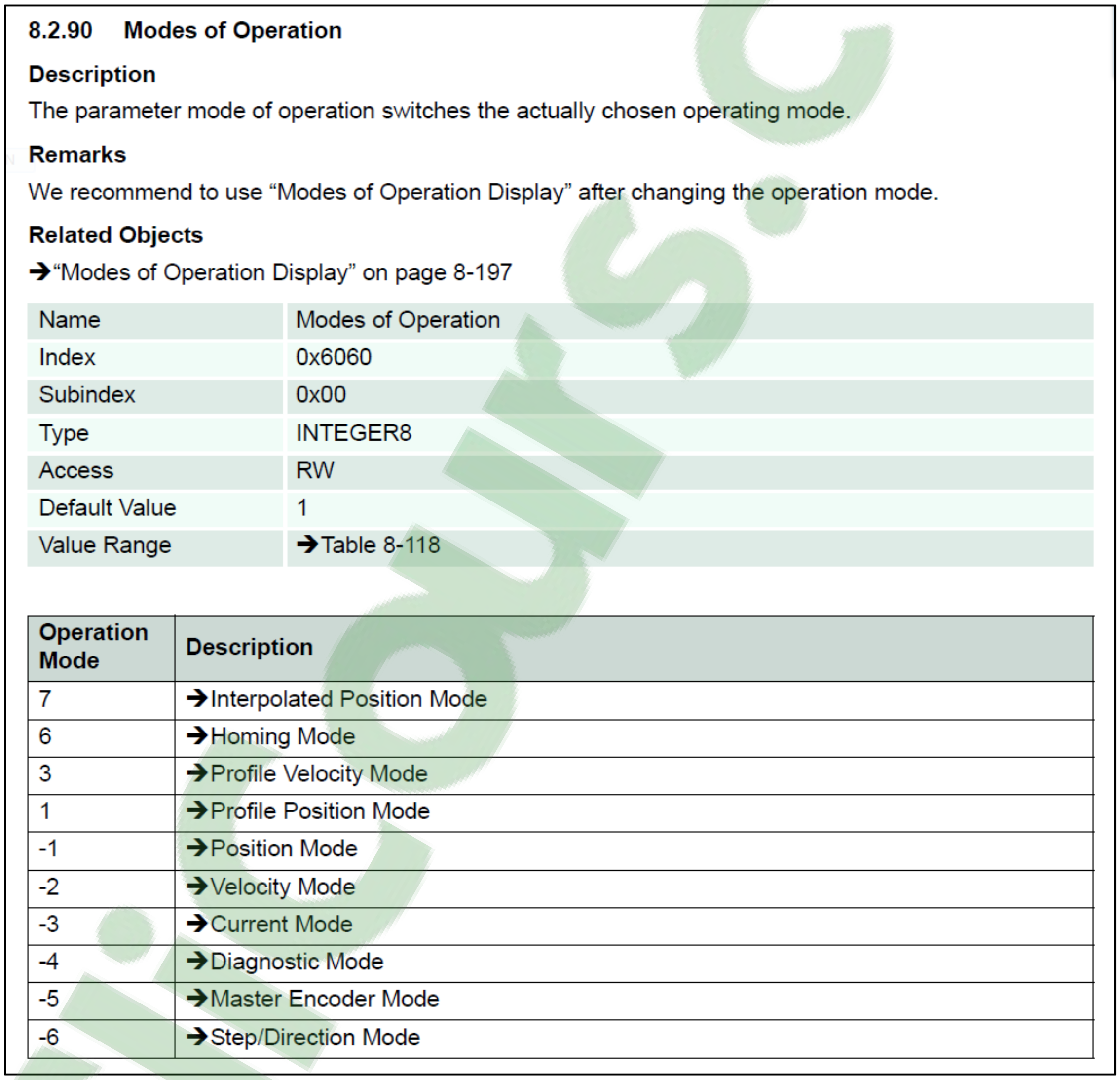
Device control commands are triggered by the following bit patterns in the ➔Controlword.

Command	LowByte of Controlword [binary]	State Transition
Shutdown	0xxx x110	2, 6, 8
Switch On	0xxx x111	3
Switch On & Enable Operation	0xxx 1111	3, 4*1)
Disable Voltage	0xxx xx0x	7, 9, 10, 12
Quickstop	0xxx x01x	7, 10, 11
Disable Operation	0xxx 0111	5
Enable Operation	0xxx 1111	4, 16
Fault Reset	0xxx xxxx → 1xxx xxxx	15

Figure-A III-3 Controlword values according to the drive state
Taken from Maxon motor control (2014c)

ANNEX IV

Modes of operation



8.2.90 Modes of Operation

Description

The parameter mode of operation switches the actually chosen operating mode.

Remarks

We recommend to use "Modes of Operation Display" after changing the operation mode.

Related Objects

➔"Modes of Operation Display" on page 8-197

Name	Modes of Operation
Index	0x6060
Subindex	0x00
Type	INTEGER8
Access	RW
Default Value	1
Value Range	➔Table 8-118

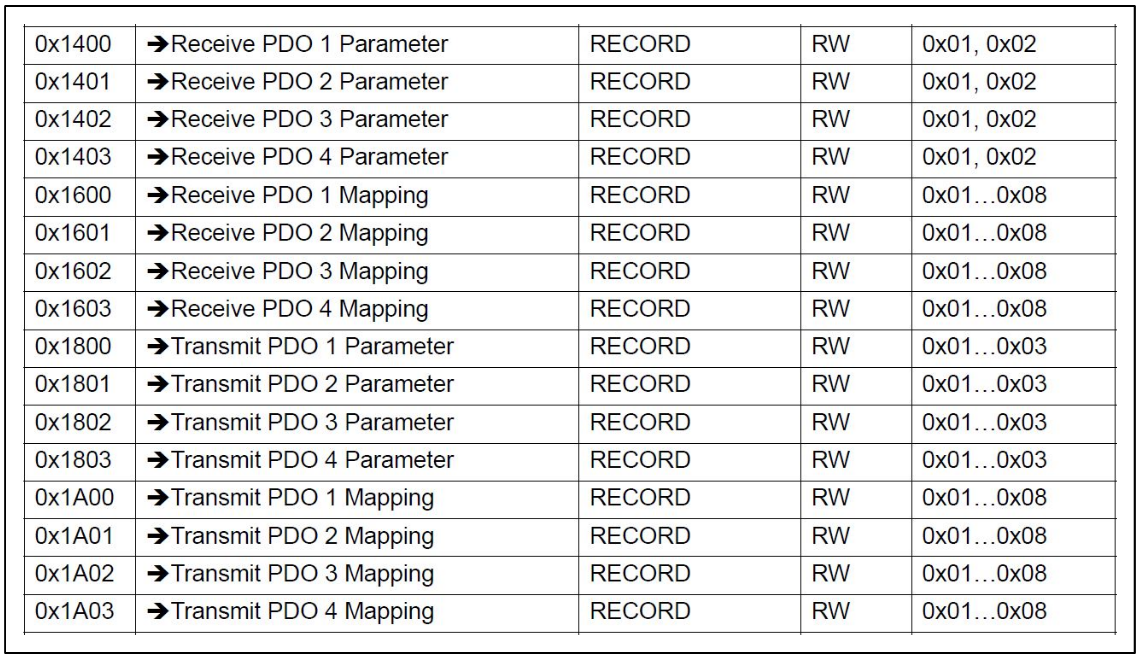
Operation Mode	Description
7	➔Interpolated Position Mode
6	➔Homing Mode
3	➔Profile Velocity Mode
1	➔Profile Position Mode
-1	➔Position Mode
-2	➔Velocity Mode
-3	➔Current Mode
-4	➔Diagnostic Mode
-5	➔Master Encoder Mode
-6	➔Step/Direction Mode

Figure-A IV-1 Modes of operation parameter description
Taken from Maxon motor control (2014c)

clicours.com

ANNEX V

RxPDOs and TxPDOs addresses



0x1400	➔Receive PDO 1 Parameter	RECORD	RW	0x01, 0x02
0x1401	➔Receive PDO 2 Parameter	RECORD	RW	0x01, 0x02
0x1402	➔Receive PDO 3 Parameter	RECORD	RW	0x01, 0x02
0x1403	➔Receive PDO 4 Parameter	RECORD	RW	0x01, 0x02
0x1600	➔Receive PDO 1 Mapping	RECORD	RW	0x01…0x08
0x1601	➔Receive PDO 2 Mapping	RECORD	RW	0x01…0x08
0x1602	➔Receive PDO 3 Mapping	RECORD	RW	0x01…0x08
0x1603	➔Receive PDO 4 Mapping	RECORD	RW	0x01…0x08
0x1800	➔Transmit PDO 1 Parameter	RECORD	RW	0x01…0x03
0x1801	➔Transmit PDO 2 Parameter	RECORD	RW	0x01…0x03
0x1802	➔Transmit PDO 3 Parameter	RECORD	RW	0x01…0x03
0x1803	➔Transmit PDO 4 Parameter	RECORD	RW	0x01…0x03
0x1A00	➔Transmit PDO 1 Mapping	RECORD	RW	0x01…0x08
0x1A01	➔Transmit PDO 2 Mapping	RECORD	RW	0x01…0x08
0x1A02	➔Transmit PDO 3 Mapping	RECORD	RW	0x01…0x08
0x1A03	➔Transmit PDO 4 Mapping	RECORD	RW	0x01…0x08

Figure-A V-1 RxPDOs and TxPDOs mapping/parameter addresses
Taken from

ANNEX VI

SDOs programming sequence for EPOS2 drives

Table-A VI 1 SDOs programming sequence for node ID ‘2’

Step	Command	Data-Type	Index(hex)	Sub-Index	Value (hex)	Comment
1	SDO Write	U16	6040	0	80	Clear all faults: Write Control Word (Reset Fault to TRUE)
2	SDO Write	U16	6040	0	0	Write Control Word (Disable)
3	SDO Write	I8	6060	0	FF	set the operation mode
4	SDO Write	U32	1800	1	182	Configure Transmit PDO1 COB-ID
5	SDO Write	U8	1800	2	1	Configure Transmit PDO1 Transmission Type to Synchronous
6	SDO Write	U8	1A00	0	0	Disable Transmit PDO1 by writing 0 to the number of mapped PDO objects
7	SDO Write	U32	1A00	1	60410010	Map the first object of TxPDO1 (StatusWord)
8	SDO Write	U32	1A00	2	60640020	Map the second object of TxPDO1(actual position value)
9	SDO Write	U8	1A00	0	2	Enable transmit TxPDO1 by writing the number of mapped objects in the field with sub index 0X00
10	SDO Write	U32	1801	1	282	Configure Transmit PDO2 COB-ID
11	SDO Write	U8	1801	2	1	Configure Transmit PDO2 Transmission Type to Synchronous
12	SDO Write	U8	1A01	0	0	Disable Transmit PDO2 by writing 0 to the number of mapped PDO objects
13	SDO Write	U32	1A01	1	20270010	Map the first object of TxPDO2(current actual value average)
14	SDO Write	U32	1A01	2	20280020	Map the second object of TxPDO2(velocity actual value average)
15	SDO Write	U8	1A01	0	2	Enable Transmit TxPDO2 by writing 2 to the number of mapped PDO objects

Table -A VI 1 SDOs programming sequence for node ID ‘2’ (Continued)

16	SDO Write	U8	1A02	0	0	Disable Transmit PDO3 by writing 0 to the number of mapped PDO objects
17	SDO Write	U8	1A03	0	0	Disable Transmit PDO4 by writing 0 to the number of mapped PDO objects
18	SDO Write	U32	1400	1	202	Configure Transmit PDO1 COB-ID
19	SDO Write	U8	1400	2	1	Configure Transmit PDO1 Transmission Type to Synchronous
20	SDO Write	U8	1600	0	0	Disable Receive PDO1 by writing 0 to the number of mapped PDO objects
21	SDO Write	U32	1600	1	60400010	Map the first object of RxPDO1 (ControlWord)
22	SDO Write	U32	1600	2	20620020	Map the second object of RxPDO1 (set target position)
23	SDO Write	U8	1600	0	2	Enable the RxPDO1 by writing the number of mapped objects in the field with sub index 0X00
24	SDO Write	U8	1601	0	0	Disable Receive PDO2 by writing 0 to the number of mapped PDO objects
25	SDO Write	U8	1602	0	0	Disable Receive PDO3by writing 0 to the number of mapped PDO objects
26	SDO Write	U8	1603	0	0	Disable Receive PDO4 by writing 0 to the number of mapped PDO objects
27	SDO Write	I8	608B	0	0	Update the velocity units to RPM
28	SDO Write	U32	60C5	0	1F40	set the limit of the acceleration (MAX acceleration)
29	SDO Write	U32	607F	0	1770	update Max profile Velocity to 6000 RPM
30	SDO Write	I32	607D	1	FFFFD120	set the min position limit to -12 000 qc
31	SDO Write	I32	607D	2	2EE0	set the max position limit to 12 000 qc

Table -A VI 2 SDOs programming sequence for node ID ‘3’

Step	Command	Data-Type	Index(hex)	Sub-Index	Value (hex)	Comment
1	SDO Write	U16	6040	0	80	Clear all faults: Write Control Word (Reset Fault to TRUE)
2	SDO Write	U16	6040	0	0	Write Control Word (Disable)
3	SDO Write	I8	6060	0	FF	set the operation mode
4	SDO Write	U32	1800	1	183	Configure Transmit PDO1 COB-ID
5	SDO Write	U8	1800	2	1	Configure Transmit PDO1 Transmission Type to Synchronous
6	SDO Write	U8	1A00	0	0	Disable Transmit PDO1 by writing 0 to the number of mapped PDO objects
7	SDO Write	U32	1A00	1	60410010	Map the first object of TxPDO1 (StatusWord)
8	SDO Write	U32	1A00	2	60640020	Map the second object of TxPDO1(actual position value)
9	SDO Write	U8	1A00	0	2	Enable transmit TxPDO1 by writing the number of mapped objects in the field with sub index 0X00
10	SDO Write	U32	1801	1	283	Configure Transmit PDO2 COB-ID
11	SDO Write	U8	1801	2	1	Configure Transmit PDO2 Transmission Type to Synchronous
12	SDO Write	U8	1A01	0	0	Disable Transmit PDO2 by writing 0 to the number of mapped PDO objects
13	SDO Write	U32	1A01	1	20270010	Map the first object of TxPDO2(current actual value average)
14	SDO Write	U32	1A01	2	20280020	Map the second object of TxPDO2(velocity actual value average)
15	SDO Write	U8	1A01	0	2	Enable Transmit TxPDO2 by writing 2 to the number of mapped PDO objects

Table -A VI 2 SDOs programming sequence for node ID ‘3’ (Continued)

16	SDO Write	U8	1A02	0	0	Disable Transmit PDO3 by writing 0 to the number of mapped PDO objects
17	SDO Write	U8	1A03	0	0	Disable Transmit PDO4 by writing 0 to the number of mapped PDO objects
18	SDO Write	U32	1400	1	203	Configure Transmit PDO1 COB-ID
19	SDO Write	U8	1400	2	1	Configure Transmit PDO1 Transmission Type to Synchronous
20	SDO Write	U8	1600	0	0	Disable Receive PDO1 by writing 0 to the number of mapped PDO objects
21	SDO Write	U32	1600	1	60400010	Map the first object of RxPDO1 (ControlWord)
22	SDO Write	U32	1600	2	20620020	Map the second object of RxPDO1 (set target position)
23	SDO Write	U8	1600	0	2	Enable the RxPDO1 by writing the number of mapped objects in the field with sub index 0X00
24	SDO Write	U8	1601	0	0	Disable Receive PDO2 by writing 0 to the number of mapped PDO objects
25	SDO Write	U8	1602	0	0	Disable Receive PDO3by writing 0 to the number of mapped PDO objects
26	SDO Write	U8	1603	0	0	Disable Receive PDO4 by writing 0 to the number of mapped PDO objects
27	SDO Write	I8	608B	0	0	Update the velocity units to RPM
28	SDO Write	U32	60C5	0	1F40	set the limit of the acceleration (MAX acceleration)
29	SDO Write	U32	607F	0	1770	update Max profile Velocity to 6000 RPM
30	SDO Write	I32	607D	1	FFFFD120	set the min position limit to -12 000 qc
31	SDO Write	I32	607D	2	2EE0	set the max position limit to 12 000 qc

Table -A VI-3 SDOs programming sequence for node ID ‘4

Step	Command	Data-Type	Index(hex)	Sub-Index	Value (hex)	Comment
1	SDO Write	U16	6040	0	80	Clear all faults: Write Control Word (Reset Fault to TRUE)
2	SDO Write	U16	6040	0	0	Write Control Word (Disable)
3	SDO Write	I8	6060	0	FF	set the operation mode
4	SDO Write	U32	1800	1	184	Configure Transmit PDO1 COB-ID
5	SDO Write	U8	1800	2	1	Configure Transmit PDO1 Transmission Type to Synchronous
6	SDO Write	U8	1A00	0	0	Disable Transmit PDO1 by writing 0 to the number of mapped PDO objects
7	SDO Write	U32	1A00	1	60410010	Map the first object of TxPDO1 (StatusWord)
8	SDO Write	U32	1A00	2	60640020	Map the second object of TxPDO1(actual position value)
9	SDO Write	U8	1A00	0	2	Enable transmit TxPDO1 by writing the number of mapped objects in the field with sub index 0X00
10	SDO Write	U32	1801	1	284	Configure Transmit PDO2 COB-ID
11	SDO Write	U8	1801	2	1	Configure Transmit PDO2 Transmission Type to Synchronous
12	SDO Write	U8	1A01	0	0	Disable Transmit PDO2 by writing 0 to the number of mapped PDO objects
13	SDO Write	U32	1A01	1	20270010	Map the first object of TxPDO2(current actual value average)
14	SDO Write	U32	1A01	2	20280020	Map the second object of TxPDO2(velocity actual value average)
15	SDO Write	U8	1A01	0	2	Enable Transmit TxPDO2 by writing 2 to the number of mapped PDO objects
16	SDO Write	U8	1A02	0	0	Disable Transmit PDO3 by writing 0 to the number of mapped PDO objects

Table -A VI-3 SDOs programming sequence for node ID ‘4 (Continued)

17	SDO Write	U8	1A03	0	0	Disable Transmit PDO4 by writing 0 to the number of mapped PDO objects
18	SDO Write	U32	1400	1	204	Configure Transmit PDO1 COB-ID
19	SDO Write	U8	1400	2	1	Configure Transmit PDO1 Transmission Type to Synchronous
20	SDO Write	U8	1600	0	0	Disable Receive PDO1 by writing 0 to the number of mapped PDO objects
21	SDO Write	U32	1600	1	60400010	Map the first object of RxPDO1 (ControlWord)
22	SDO Write	U32	1600	2	20620020	Map the second object of RxPDO1 (set target position)
23	SDO Write	U8	1600	0	2	Enable the RxPDO1 by writing the number of mapped objects in the field with sub index 0X00
24	SDO Write	U8	1601	0	0	Disable Receive PDO2 by writing 0 to the number of mapped PDO objects
25	SDO Write	U8	1602	0	0	Disable Receive PDO3by writing 0 to the number of mapped PDO objects
26	SDO Write	U8	1603	0	0	Disable Receive PDO4 by writing 0 to the number of mapped PDO objects
27	SDO Write	I8	608B	0	0	Update the velocity units to RPM
28	SDO Write	U32	60C5	0	1F40	set the limit of the acceleration (MAX acceleration)
29	SDO Write	U32	607F	0	1770	update Max profile Velocity to 6000 RPM
30	SDO Write	I32	607D	1	FFFFD120	set the min position limit to -12 000 qc
31	SDO Write	I32	607D	2	2EE0	set the max position limit to 12 000 qc

Table -A VI-4 SDOs programming sequence for node ID ‘5’

Step	Command	Data-Type	Index(hex)	Sub-Index	Value (hex)	Comment
1	SDO Write	U16	6040	0	80	Clear all faults: Write Control Word (Reset Fault to TRUE)
2	SDO Write	U16	6040	0	0	Write Control Word (Disable)
3	SDO Write	I8	6060	0	FF	set the operation mode
4	SDO Write	U32	1800	1	185	Configure Transmit PDO1 COB-ID
5	SDO Write	U8	1800	2	1	Configure Transmit PDO1 Transmission Type to Synchronous
6	SDO Write	U8	1A00	0	0	Disable Transmit PDO1 by writing 0 to the number of mapped PDO objects
7	SDO Write	U32	1A00	1	60410010	Map the first object of TxPDO1 (StatusWord)
8	SDO Write	U32	1A00	2	60640020	Map the second object of TxPDO1(actual position value)
9	SDO Write	U8	1A00	0	2	Enable transmit TxPDO1 by writing the number of mapped objects in the field with sub index 0X00
10	SDO Write	U32	1801	1	285	Configure Transmit PDO2 COB-ID
11	SDO Write	U8	1801	2	1	Configure Transmit PDO2 Transmission Type to Synchronous
12	SDO Write	U8	1A01	0	0	Disable Transmit PDO2 by writing 0 to the number of mapped PDO objects
13	SDO Write	U32	1A01	1	20270010	Map the first object of TxPDO2(current actual value average)
14	SDO Write	U32	1A01	2	20280020	Map the second object of TxPDO2(velocity actual value average)
15	SDO Write	U8	1A01	0	2	Enable Transmit TxPDO2 by writing 2 to the number of mapped PDO objects
16	SDO Write	U8	1A02	0	0	Disable Transmit PDO3 by writing 0 to the number of mapped PDO objects

ClicCours.com

Table -A VI-4 SDOs programming sequence for node ID ‘5’ (Continued)

17	SDO Write	U8	1A03	0	0	Disable Transmit PDO4 by writing 0 to the number of mapped PDO objects
18	SDO Write	U32	1400	1	205	Configure Transmit PDO1 COB-ID
19	SDO Write	U8	1400	2	1	Configure Transmit PDO1 Transmission Type to Synchronous
20	SDO Write	U8	1600	0	0	Disable Receive PDO1 by writing 0 to the number of mapped PDO objects
21	SDO Write	U32	1600	1	60400010	Map the first object of RxPDO1 (ControlWord)
22	SDO Write	U32	1600	2	20620020	Map the second object of RxPDO1 (set target position)
23	SDO Write	U8	1600	0	2	Enable the RxPDO1 by writing the number of mapped objects in the field with sub index 0X00
24	SDO Write	U8	1601	0	0	Disable Receive PDO2 by writing 0 to the number of mapped PDO objects
25	SDO Write	U8	1602	0	0	Disable Receive PDO3by writing 0 to the number of mapped PDO objects
26	SDO Write	U8	1603	0	0	Disable Receive PDO4 by writing 0 to the number of mapped PDO objects
27	SDO Write	I8	608B	0	0	Update the velocity units to RPM
28	SDO Write	U32	60C5	0	1F40	set the limit of the acceleration (MAX acceleration)
29	SDO Write	U32	607F	0	1770	update Max profile Velocity to 6000 RPM
30	SDO Write	I32	607D	1	FFFFD120	set the min position limit to -12 000 qc
31	SDO Write	I32	607D	2	2EE0	set the max position limit to 12 000 qc

ANNEX VII

Detailed results for skin displacements control using Lookup tables

Table-A VII-1

	Actuator 1 [mm]			**Actuator 2 [mm]**			**Actuator 3 [mm]**			**Actuator 4 [mm]**		
Flight case	**Theoretical displacement**	**DDI**	**error**	**Theoretical displacement**	**DDI**	**error**	**Theoretical displacement**	**DDI**	**error**	**Theoretical displacement**	**DDI**	**error**
1	-2.26	-2.3	0.04	-1.66	-1.7	0.04	-2.09	-1.99	0.1	-1.54	-1.52	0.02
2	-1.86	-1.9	0.04	-0.55	-0.69	0.14	-1.73	-1.72	0.01	-0.51	-0.58	0.07
3	-2.32	-2.36	0.04	-1.74	-1.79	0.05	-2.15	-2.07	0.08	-1.61	-1.53	0.08
4	-2.1	-2.13	0.03	-1.38	-1.42	0.04	-1.95	-1.98	0.03	-1.28	-1.13	0.15
5	-2.12	-2.17	0.05	-1.48	-1.53	0.05	-1.97	-1.87	0.1	-1.37	-1.26	0.11
6	-1.97	-1.96	0.01	-1.19	-1.21	0.02	-1.83	-1.84	0.01	-1.1	-0.91	0.19
7	-1.96	-1.95	0.01	-1.57	-1.64	0.07	-1.81	-1.81	0	-1.46	-1.38	0.08
8	-1.69	-1.75	0.06	-1.48	-1.57	0.09	-1.57	-1.68	0.11	-1.37	-1.33	0.04
9	-0.8	-0.91	0.11	-1.17	-1.2	0.03	-0.82	-0.81	0.01	-1.09	-1.04	0.05
10	-2.09	-2.13	0.04	2.86	2.79	0.07	-1.94	-2.05	0.11	2.65	2.6	0.05
11	2.7	2.71	0.01	-2.43	-2.42	0.01	2.5	2.32	0.18	-2.26	-2.27	0.01
12	-2.06	-2.12	0.06	2.85	2.77	0.08	-1.91	-1.88	0.03	2.65	2.59	0.06
13	-1.56	-1.62	0.06	2.91	2.86	0.05	-1.45	-1.56	0.11	2.75	2.72	0.03
14	-2.56	-2.62	0.06	2.6	2.56	0.04	-2.38	-2.46	0.08	2.42	2.43	0.01
15	2.85	2.86	0.01	-2.51	-2.53	0.02	2.65	3.16	0.17	-2.33	-2.31	0.02

Table A VII-1 (Continued)

	Actuator 1 [mm]			**Actuator 2 [mm]**			**Actuator 3 [mm]**			**Actuator 4 [mm]**		
Flight case	**Theoretical displacement**	**DDI**	**error**	**Theoretical displacement**	**DDI**	**error**	**Theoretical displacement**	**DDI**	**error**	**Theoretical displacement**	**DDI**	**error**
16	-2.12	-2.15	0.03	1.92	1.89	0.03	-1.96	-1.91	0.05	1.78	1.73	0.05
17	-1.81	-1.91	0.1	2.69	2.64	0.05	-1.68	-1.73	0.05	2.5	2.55	0.05
18	2.89	2.87	0.02	-1.11	-1.15	0.04	2.69	2.52	0.17	-1.03	-1	0.03
19	-2.99	-3.05	0.06	2.63	2.62	0.01	-2.77	-2.91	0.14	2.44	2.5	0.06
20	-0.99	-1.02	0.03	1.85	1.78	0.07	-0.92	-1.04	0.12	1.72	1.65	0.07
21	-1.57	-1.59	0.02	2.46	2.38	0.08	-1.46	-1.54	0.08	2.28	2.31	0.03
22	-3.05	-3	0.05	2.46	2.45	0.01	-2.83	-2.94	0.11	2.28	2.33	0.05
23	-2.02	-2.04	0.02	2.59	2.53	0.06	-1.87	-1.97	0.1	2.4	2.38	0.02
24	-2.07	-2.11	0.04	2.96	2.89	0.07	-1.92	-2.05	0.13	2.74	2.67	0.07
25	-2.5	-2.48	0.02	1.78	1.77	0.01	-2.32	-2.4	0.08	1.65	1.63	0.02
26	-1.39	-1.38	0.01	2.78	2.7	0.08	-1.29	-1.41	0.12	2.58	2.59	0.01
27	-2.67	-2.72	0.05	1.54	1.54	0	-2.48	-2.58	0.1	1.43	1.42	0.01
28	-1.51	-1.5	0.01	2.43	2.34	0.09	-1.4	-1.51	0.11	2.26	2.26	0
29	-2.49	-2.53	0.04	2.24	2.22	0.02	-2.31	-2.42	0.11	2.08	2.08	0
30	-2.51	-2.58	0.07	2.99	2.94	0.05	-2.32	-2.47	0.15	2.77	2.72	0.05

ANNEX VII

Kollmorgen® drive parameters

Tableau-A VI-1 Used parameters list for Modbus communication
Adapted from Kollmorgen (2014)

Parameter name	Modbus register address (Decimal addressing)	Description
MODBUS.DRVSTAT	944	Returns drive status, limit hardware and software switches, etc...
MODBUS.DRV	942	Used to enable, disable or stop the drive
DRV.CMDSOURCE	226	Sets the command source (service, fieldbus, analog input, etc…)
DRV.OPMODE	270	Sets the drive operation mode(current, velocity, or position)
HOME.SET	418	Immediately sets the home position; active in position mode only
MT.MOVE	544	Starts a motion task; active in position mode only
DRV.MOTIONSTAT	268	Reads the motion status of the drive

ANNEX IX

Actuators displacement profiles

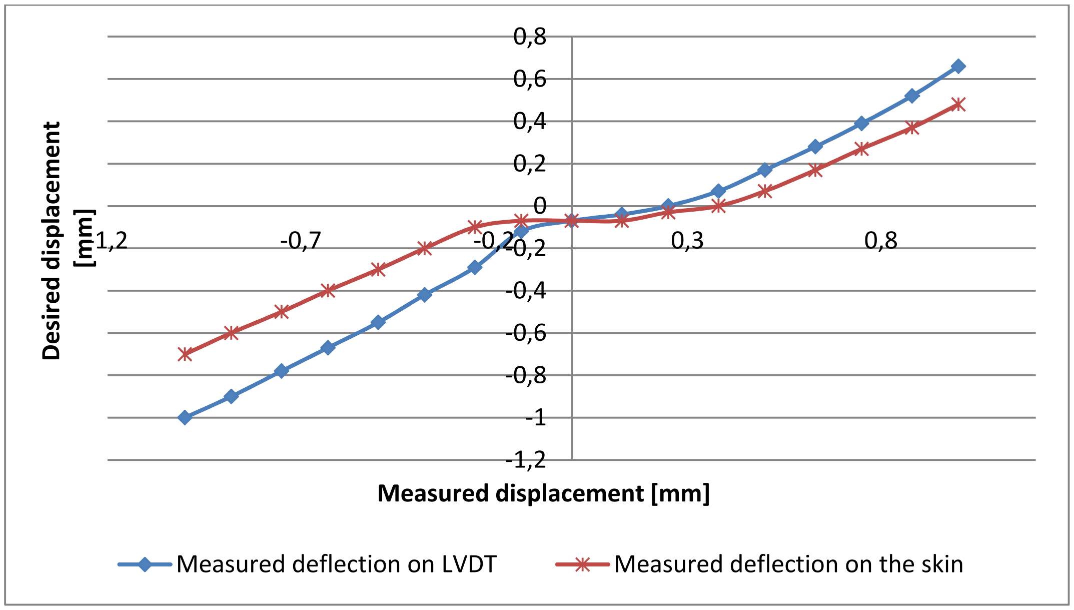


Figure-A IX-1 displacement profile for actuator ‘2’

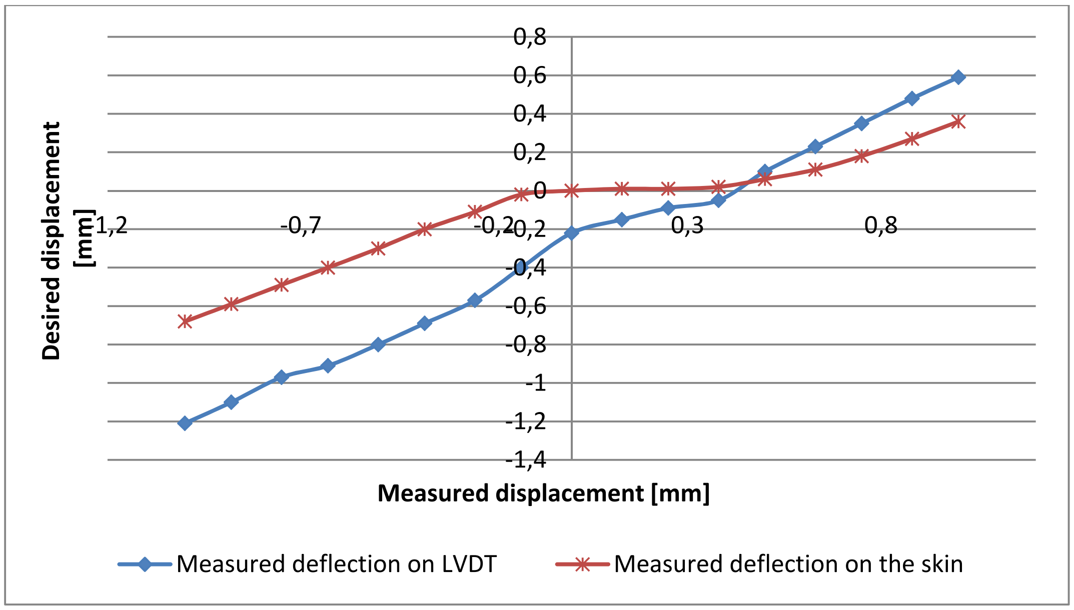


Figure-A IX-2 displacement profile for actuator '3'

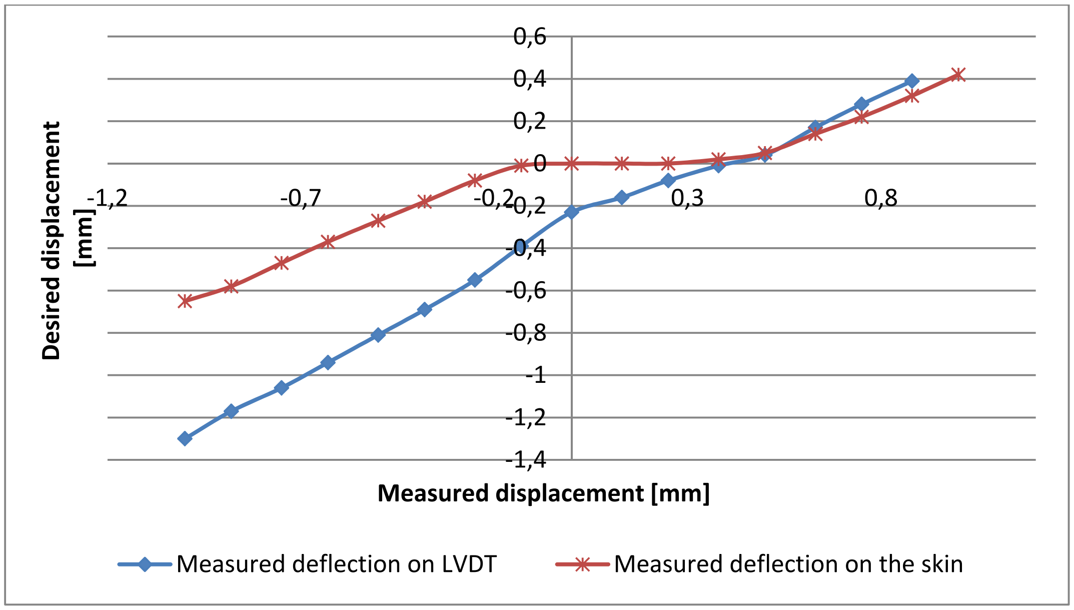


Figure-A IX-3 displacement profile for actuator '4'

ANNEX X

Aerodynamic comparison

Tableau-A X-1 Comparison of the transition for the non-morphed wing during the first test campaign

Test case	IR method transition	Kulite method transition	Fluent-predicted transition	Absolute error (IR-Fluent)	Absolute error (IR-Kulite)
1	56,18%	53,45%	66,00%	9,82%	2,72%
2	73,35%	51,20%	65,00%	8,35%	22,15%
3	73,30%	51,20%	64,00%	9,30%	22,10%
4	57,36%	50,04%	62,00%	4,64%	7,32%
5	73,18%	46,28%	63,00%	10,18%	26,90%
6	73,51%	46,28%	62,00%	11,51%	27,23%
7	56,05%	46,28%	62,00%	5,95%	9,77%
8	53,79%	46,28%	60,00%	6,21%	7,50%
9	53,17%	46,28%	59,00%	5,83%	6,89%
10	73,36%	63,81%	77,00%	3,64%	9,55%
11	69,47%	63,81%	77,00%	7,53%	5,66%
12	66,62%	46,28%	75,00%	8,38%	20,34%
13	65,99%	38,25%	74,00%	8,01%	27,74%
14	65,92%	67,72%	74,00%	8,08%	1,80%
15	67,77%	67,72%	74,00%	6,23%	0,06%
16	67,13%	63,81%	69,00%	1,87%	3,32%
17	68,14%	63,81%	68,00%	0,14%	4,33%
18	65,70%	38,25%	67,00%	1,30%	27,45%
19	65,28%	58,75%	66,00%	0,72%	6,52%
20	60,12%	50,79%	65,00%	4,88%	9,33%
21	56,97%	50,79%	64,00%	7,03%	6,18%
22	55,20%	50,79%	63,00%	7,80%	4,41%
23	54,75%	46,28%	62,00%	7,25%	8,47%
24	52,02%	46,28%	60,00%	7,98%	5,73%
25	52,02%	45,16%	58,00%	5,98%	6,86%

Tableau-A X-2 Comparison of the transition for the morphed wing during the first test campaign

Test case	IR method transition	Kulite method transition	Fluent-predicted transition	Absolute error (IR-Fluent)	Absolute error (IR-Kulite)
1	53,57%	46,28%	67,00%	13,43%	7,29%
2	73,41%	46,28%	66,00%	7,41%	27,13%
3	73,46%	45,16%	64,00%	9,46%	28,30%
4	72,97%	45,16%	64,00%	8,97%	27,81%
5	73,50%	45,16%	63,00%	10,50%	28,35%
6	73,37%	45,16%	62,00%	11,37%	28,21%
7	50,66%	45,16%	61,00%	10,34%	5,50%
8	50,39%	46,28%	60,00%	9,61%	4,11%
9	52,07%	46,28%	59,00%	6,93%	5,78%
10	58,00%	46,28%	47,00%	11,00%	11,72%
11	65,73%	58,75%	62,00%	3,73%	6,98%
12	55,67%	46,28%	48,00%	7,67%	9,38%
13	56,77%	50,79%	49,00%	7,77%	5,98%
14	54,11%	46,28%	47,00%	7,11%	7,83%
15	63,11%	58,75%	63,00%	0,11%	4,36%
16	61,86%	50,79%	48,00%	13,86%	11,06%
17	60,38%	46,28%	47,00%	13,38%	14,09%
18	63,94%	57,21%	60,00%	3,94%	6,73%
19	52,55%	44,36%	50,00%	2,55%	8,19%
20	54,89%	46,28%	55,00%	0,11%	8,60%
21	51,84%	46,28%	48,00%	3,84%	5,56%
22	48,93%	44,36%	44,00%	4,93%	4,57%
23	49,55%	44,36%	45,00%	4,55%	5,19%
24	48,50%	44,36%	45,00%	3,50%	4,14%
25	48,97%	43,62%	47,00%	1,97%	5,35%

LIST OF REFERENCES

Alena, R.L., J.P. Ossenfort IV, K. Laws, A. Goforth et F. Figueroa. 2007. « Communications for Integrated Modular Avionics ». In *Aerospace Conference, 2007 IEEE*. p. 1-18. IEEE.

Åström, K.J. 2002. « Control System Design : lecture notes for me 155a ». *Department of Mechanical and Environmental Engineering University of California Santa Barbara*. p. 333.

Ayre, O.P.A., et Ch. Keydel. 2003. *Embedded Networking with CAN and CANopen*. the United States of America: RTC Books.

Barbarino, S., O. Bilgen, R.M. Ajaj, M.I. Friswell et D.J. Inman. 2011. « A review of morphing aircraft ». *Journal of Intelligent Material Systems and Structures,* vol. 22, n° 9, p. 823-877.

Blevins, T.L., et M. Nixon. 2010. *Control Loop Foundation: Batch and Continuous Processes*. ISA.

Bowman, J., B. Sanders et T. Weisshaar. 2002. « Evaluating the Impact of Morphing Technologies on Aircraft Performance ». *AIAA paper,* vol. 1631, p. 2002.

CAN in automation. 2011. *301 V4. 2.0–CANopen application layer and communication profile, CAN in Automation* V4. 2.0.

Chopra, I. 2002. « Review of State of Art of Smart Structures and Integrated Systems ». *AIAA journal,* vol. 40, n° 11, p. 2145-2187.

Dietrich, D., et T. Sauter. 2000. « Evolution Potentials for Fieldbus Systems ». In *Factory Communication Systems, 2000. Proceedings. 2000 IEEE International Workshop on*. (6-8 Sept. 2000), p. 343.

Gorez, R. 1997. « A survey of PID auto-tuning methods: Feature issue controller tuning ». *Journal A,* vol. 38, n° 1, p. 3-10.

Green Aviation. 2010. « A Better Way to treat the Planet ». *National Aeronautics and Space Administration, NF-2010-07-500-HQ.*

Inman, D.J. 2001. « Wings: Out of the Box. Determining Actuator Requirements for Controlled Morphing Air Vehicles-Aerodynamic Loads ». In *DARPA Technology Interchange Meeting, Wright Patterson Air Force Base, Dayton, OH.*

clicours.com

Ioan D.L., et Z. Gianluca. 2006. *Digital Control Systems*. London: Springer-Verlag London Limited 2006.

Knospe, C. 2006. « PID control ». *IEEE Control Systems Magazine,* vol. 26, n^{o} 1, p. 30-31.

Kollmorgen. 2014. « AKD User Guide ». Decembre 2014, Revision N. < http://www.kollmorgen.com/en-us/products/drives/servo/akd/_manuals/kollmorgen-akd-user-guide-en-rev-n/ >. Consulté le 12 septembre 2015.

Koreanschi, A., O. Sugar Gabor et R.M Botez. 2015. « Numerical and Experimental Validation of a Morphed Wing Geometry using Price-Paidoussis Wind Tunnel Testing ». *The Aeronautical Journal, Accepted for publication*.

Mamou, M, Y Mébarki, M Khalid, M Genest, D Coutu, AV Popov, C Sainmont, T Georges, L Grigorie et RM Botez. 2010. « Aerodynamic performance optimization of a wind tunnel morphing wing model subject to various cruise flow conditions ». *Proc of ICAS'27*, p. 1-18.

Maxon motor control. 2013a. « EPOS2 Positioning Controller-Getting Started ». < http://www.maxonmotor.com/medias/sys_master/8803613966366/323232-Getting-Started-En.pdf >. Consulté le 3 septembre 2015.

Maxon motor control. 2013b. « EPOS2 Positioning Controllers-Communication Guide ». p. 54. < http://www.maxonmotor.com/medias/sys_master/8806425067550/EPOS2-Communication-Guide-En.pdf >. Consulté le 15 septembre 2015.

Maxon motor control. 2014a. « EPOS2 24/5 Positioning Controller-Hardware Reference ». < http://www.maxonmotor.com/medias/sys_master/8815046721566/378308-Hardware-Reference-En.pdf >. Consulté le 10 octobre 2015.

Maxon motor control. 2014b. « EPOS2 Positioning Controllers-Application Notes Collection ». < http://www.maxonmotor.nl/medias/sys_master/8811457937438/EPOS2-Application-Notes-Collection-En.pdf >. Consulté le 2 octobre 2015.

Maxon motor control. 2014c. « EPOS2 Positioning Controllers-Firmware Specification ». < http://www.maxonmotorusa.com/medias/sys_master/8815045967902/EPOS2-Firmware-Specification-En.pdf >. Consulté le 23 septembre 2015.

Maxon motor control. 2015. « EPOS2 24/5, Digital positioning controller, 5 A, 11 - 24 VDC ». < http://www.maxonmotor.com/maxon/view/product/control/Positionierung/367676 >. Consulté le 15 octobre 2015.

Mitutoyo. 2013. *USB-ITPAK User's Manual*.

Modbus Organization Inc. 2006. *MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b*.

Modbus Organization, Inc. 2015. « Modbus FAQ: About The Modbus Organization ». < http://www.modbus.org/faq.php >. Consulté le 18/09/2015.

Mujahid, A., et R. Lind. 2006. « Using avian morphology to enhance aircraft maneuverability ». In *AIAA Atmospheric Flight Mechanics Conference and Exhibit*. p. 6643.

National Instruments. 2015. « Components of a project ». < http://zone.ni.com/reference/en-XX/help/372846J-01/veristand/comp_of_project/ >. Consulté le 30 August 2014.

National Research Council Canada. 2013. « 2 m x 3 m wind tunnel ». < http://www.nrc-cnrc.gc.ca/eng/solutions/facilities/wind_tunnel/2x3_metre.html >. Consulté le 16 juin 2015.

Popov, A.V. 2010. « Design of an Active Controller for Delaying the Transition from Laminar Flow to Turbulent Flow over a Morphing Wing in Wind Tunnel ». École de technologie supérieure.

Popov, A.V., R.M. Botez, M. Mahmoud, M. Youssef, B. Jahraus, M. Khaled et T.L. Grigorie. 2009. « Drag Reduction by Improving Laminar Flows Past Morphing Configurations ». In *AVT-168 NATO Symposium on the Morphing Vehicles*. (Evora, Portugal).

Popov, A.V., T.L. Grigorie, R. M. Botez, Y. Mébarki et M. Mamou. 2010. « Modeling And Testing of a Morphing Wing in Open-loop Architecture ». *Journal of Aircraft,* vol. 47, n^{o} 3, p. 917-923.

Popov, A.V., T.L. Grigorie et R.M. Botez. 2009. « Control of a Morphing Wing in Bench Tests ». In *CASI Aircraft Design and Development Symposium*. (Kanata, Ont).

Schmidt, Kai. 2012. « The significance of EDS files in daily work with CANopen devices ». < http://vector.com/portal/medien/cmc/application_notes/AN-ION-1-1105_Meaning_of_EDS_Files_for_CANopen_Devices.pdf >. Consulté le 10 octobre 2015.

Stanewsky, E. 2001. « Adaptive Wing and Flow Control Technology ». *Progress in Aerospace Sciences,* vol. 37, n^{o} 7, p. 583-667.

Sugar Gabor, O., A. Koreanschi et R. M. Botez. 2013. « Optimization of an Unmanned Aerial Systems' Wing Using a Flexible Skin Morphing Wing ». *SAE International Journal of Aerospace,* vol. 61, p. 115-121.

Sugar Gabor, O., A. Koreanschi et R.M Botez. 2012. « Low-speed Aerodynamic Characteristics Improvement of ATR 42 Airfoil Using a Morphing Wing Approach ». In *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*. p. 5451-5456. IEEE.

Sugar Gabor, O., A. Koreanschi et R.M Botez. 2015. « Analysis of the UAS-S4 Éhecatl Airfoil High Angles of Attack Performance Characteristics Using a Morphing Wing Approach ». *Proceedings of the Institute of Mechanical Engineers – Part G: Journal of Aerospace Engineering*.

Sugar Gabor, O., A. Simon, A. Koreanschi et R.M Botez. 2015. « Aerodynamic Performance Improvement of the UAS-S4 Éhecatl Morphing Airfoil Using Novel Optimization Techniques ». *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*.

Sunit K . Sen. 2014a. « CAN Bus ». In *Fieldbus and Networking in Process Automation*. p. 199-206. CRC Press. < http://dx.doi.org/10.1201/b16891-11 >. Consulté le 2015/10/16.

Sunit K . Sen. 2014b. « MODBUS and MODBUS PLUS ». In *Fieldbus and Networking in Process Automation*. p. 185-197. CRC Press. < http://dx.doi.org/10.1201/b16891-10 >. Consulté le 2015/10/18.

Swales, Andy. 1999. « Open Modbus/TCP Specification ». *Schneider Electric,* vol. 29.

Thill, C, J Etches, I Bond, K Potter et P Weaver. 2008. « Morphing Skins ». *The Aeronautical Journal,* vol. 112, n^o 1129, p. 117-139.

Thomesse, J. P. 2005. « Fieldbus Technology in Industrial Automation ». *Proceedings of the IEEE,* vol. 93, n^o 6, p. 1073-1101.

TTTch Computertechnik. 2005. *Protocols for Aerospace Control Systems : A comparison of AFDX, aRINC 429, CAN and TTP*.

Wang, Qing-Guo. 2005. *Handbook of PI and PID Controller Tuning Rules, Aidan O'Dwyer, Imperial College Press, London, 375pp, ISBN 1-86094-342-X, 2003*. Pergamon.

Weisshaar, T.A. 2006. *Morphing Aircraft Technology-New Shapes for Aircraft Design*. DTIC Document.

Wereley, N.M, et F. Gandhi. 2010. « Special Issue Flexible Skins for Morphing Aircraft ». *Journal of Intelligent Material Systems and Structures* vol. 21

Wescott, Tim. 2000. « PID without a PhD ». *Embedded Systems Programming,* vol. 13, n^{o} 11, p. 1-7.

WIKIPEDIA. 2015a. « Fast Fourier transform ». < https://en.wikipedia.org/wiki/Fast_Fourier_transform >. Consulté le 14 July 2015.

WIKIPEDIA. 2015b. « Linear Variable Differential Transformer ». < https://en.wikipedia.org/wiki/Linear_variable_differential_transformer >. Consulté le 20 August 2015.

WIKIPEDIA. 2015c. « Standard deviation ». < https://en.wikipedia.org/wiki/Standard_deviation >. Consulté le 11 octobre 2015.