Automatic Fusion of Fidelity sources of Aerodynamic Data

Simulating Aircraft Stability And Control Characteristics for Use in Conceptual Design

Sofia M.N Chevrolat
June 3rd, 2010 - December 3rd, 2010

Master’s Thesis at the Aeronautical and Vehicle Engineering Department
Supervisor: Maximilian Tomac
Examiner: Arthur Rizzi

TRITA AVE 2011-09
ISSN 1651-7660
Useful information

**ENSPS**
Pôle API - Parc d'innovation
Bd Sébastien Brant
BP 10413
67412 Illkirch cedex
FRANCE

**Academic dean**
Mr Pierre Graebling
Tel: +33 3 68 85 44 95
Fax:+33 3 68 85 43 56
Email:pierre.graebling@unistra.fr

**Adviser**
Mrs Mireille Louys
Tel:+33 3 90 24 24 34
Fax:+33 3 90 24 24 35
Email:mireille.louys@unistra.fr

**KTH**
Royal Institute of Technology
Valhallavägen 79
11427 Stockholm
SWEDEN

**Head of division**
Pr Arthur Rizzi
Tel: +46 8 790 76 00
Fax: +46 8 20 76 20
Email: rizzi@kth.se

**Tutor**
Mr Maximilian Tomac
Tel:+46 8 790 64 67
Fax:+46 8 20 78 65
Email:maxtomac@kth.se
Acknowledgements

I would like to address special thanks to Professor Arthur Rizzi for giving me the great opportunity to carry out my master’s thesis in his department, working on a very interesting project, as well as to my tutor Mr Maximilian Tomac ”Mio” who was always very friendly and helpful, and to Mr Jesper Oppelstrup, who, several times, helped shed a reveling light on delicate issues. I would also like to thank the whole Aeronautical and Engineering department, and especially Meng Meng and Luis, for the friendly atmosphere that has reigned throughout the 6 months I was there.

Finally, a big hearty ”thank you” to Emire Lowensky, Samantha Chevrolat, and Véronique Noury, without whom this experience would hardly have been possible.
Abstract

Automatic Fusion of Fidelity Sources of Aerodynamic Data

CFD use has increased significantly in airplane conception, and the industry demands more and more precise and reliable tools. This was the goal of the SimSAC project. The result is CEASIOM, a computerized environment made of several modules for the design and prediction of the aircraft’s characteristics. It constructs aerodynamic tables used in the prediction of the characteristics of an aircraft. In simple flight conditions, simple computation methods are used, whereas in complex flight conditions, involving turbulences, more advanced methods are used. This reduces the computational cost, but the tables resulting from different fidelity sources must be fused to obtain a coherent table covering the whole flight envelope.

The goal of this project was to realize the fusion. Additionally, a filter and a custom-made mapping to enhance the accuracy of the results from the fusion were required. The addition of helpful visualization tools was suggested. The whole should be integrated in the CEASIOM interface as a Fusion module. For this, 6 functions were coded. The first one loads the data sets. The second, myplot, allows the engineer by plotting the data in a coherent way, to spot any big mistakes or incompatibility in the data sets. The third, myvisual, displays the elements spotted as outliers or potentially out of pattern. This is used by the next function, myfiltermap, to filter out the erroneous data. This function also realizes the custom-made mapping. The fifth function, myfusion, fuses the data and saves it in a .xml CEASIOM formatted structure to be used by the next CEASIOM module. The sixth function filters out, in the same way as myfiltermap, the outliers from the fused data, and saves the filtered fused data set in a .xml CEASIOM formatted structure. Finally, a Matlab GUI was implemented and integrated into the main CEASIOM interface.

The module works perfectly, except for the mapping part, that needs a few readjustments.
**Résumé**

**Fusion automatique de données aérodynamiques à fidélité variable**

L'emploi de la Mécanique des Fluides Numérique a augmenté significativement dans le domaine de la conception aéronautique, et l'industrie demande des outils de plus en plus fiables et précis. C'est le but du projet SimSAC. Le résultat est CEASIOM, un environnement programmatique composé de différents modules pour le design et la prédiction des caractéristiques d’un avion. Il construit des tables dites "aérodynamiques", utilisées dans la prédiction des caractéristiques d’un appareil. Pour les conditions de vol simples, des méthodes simplificatrices sont utilisées, alors que pour des conditions de vol complexes, incluant par exemple des turbulences, des méthodes plus avancées sont utilisées. Cela réduit le temps de calcul, mais les tables aérodynamiques résultant des différentes méthodes doivent être fusionnées afin d’obtenir une table cohérente couvrant l’ensemble du domaine de vol.

Le but de ce projet était de réaliser la fusion. De plus, un filtre et une projection sur-mesure des données, afin d’améliorer l’exactitude des résultats provenant de la fusion, étaient requis. L’addition d’outils de visualisation pertinents fut suggérée. L’ensemble devait être intégré à CEASIOM, comme le module Fusion. Pour cela, 6 fonctions ont été codées. La première charge les sets de données. La seconde, myplot, permet à l’ingénieur, en traçant les données d’une façon cohérente, de détecter des erreurs ou des incompatibilités dans les sets de données. La troisième, myvisual, affiche les points considérés comme aberrants ou potentiellement aberrants. La quatrième, myfiltermap, réalise le filtrage ainsi qu’une projection sur-mesure des données. La cinquième fonction, myfusion, fuse les données et enregistre l’ensemble dans une structure .xml au format utilisé par CEASIOM afin d’être utilisé par le prochain module. La sixième fonction filtre les points aberrants se trouvant dans les données fusionnées. Enfin, une interface graphique Matlab intégrée à l’interface principale de CEASIOM fut créée.

Le module fonctionne parfaitement, mis à part la partie projection, qui nécessite quelques réajustements.
# Contents

I  
I.1  
I.2  
I.3  
I.4  
I.5  

II  
II.1  
II.2  
II.3  
II.4  

III  
III.1  
III.2  
III.3  

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.1 Kungliga Tekniska Högskolan (KTH) - The Royal Institute of Technology</td>
<td>9</td>
</tr>
<tr>
<td>I.2 Basic Aerodynamics notions</td>
<td>10</td>
</tr>
<tr>
<td>I.2.1 Aerodynamic Forces and Moments</td>
<td>10</td>
</tr>
<tr>
<td>I.2.2 Anatomy of an aircraft</td>
<td>11</td>
</tr>
<tr>
<td>I.3 Parameters of importance</td>
<td>13</td>
</tr>
<tr>
<td>I.4 Computational Fluid Dynamics</td>
<td>14</td>
</tr>
<tr>
<td>I.4.1 General points</td>
<td>14</td>
</tr>
<tr>
<td>I.4.2 Methodology</td>
<td>16</td>
</tr>
<tr>
<td>I.4.3 Applications</td>
<td>17</td>
</tr>
<tr>
<td>I.5 The SimSAC project</td>
<td>18</td>
</tr>
<tr>
<td>I.5.1 Presentation</td>
<td>17</td>
</tr>
<tr>
<td>I.5.2 Motivation</td>
<td>19</td>
</tr>
<tr>
<td>I.5.3 CEASIOM</td>
<td>19</td>
</tr>
<tr>
<td>II.1 Presentation and Motivations</td>
<td>21</td>
</tr>
<tr>
<td>II.2 Scope of work</td>
<td>22</td>
</tr>
<tr>
<td>II.3 Principle: kriging — an interpolation process</td>
<td>22</td>
</tr>
<tr>
<td>II.4 Tools</td>
<td>23</td>
</tr>
<tr>
<td>II.4.1 The DACE Toolbox</td>
<td>23</td>
</tr>
<tr>
<td>II.4.2 the XML Toolbox</td>
<td>24</td>
</tr>
<tr>
<td>II.4.3 GUIDE: Matlab’s tool for GUI creation</td>
<td>25</td>
</tr>
<tr>
<td>II.4.4 Suplabel Toolbox</td>
<td>25</td>
</tr>
<tr>
<td>III.1 Data visualization</td>
<td>26</td>
</tr>
<tr>
<td>III.1.1 myplot</td>
<td>26</td>
</tr>
<tr>
<td>III.1.1.1 Purpose</td>
<td>26</td>
</tr>
<tr>
<td>III.1.1.2 Code description</td>
<td>26</td>
</tr>
<tr>
<td>III.1.1.3 Output</td>
<td>29</td>
</tr>
<tr>
<td>III.1.1.4 Examples of use</td>
<td>31</td>
</tr>
<tr>
<td>III.1.2 myvisual</td>
<td>32</td>
</tr>
<tr>
<td>III.1.2.1 Purpose</td>
<td>32</td>
</tr>
<tr>
<td>III.1.2.2 Code description</td>
<td>33</td>
</tr>
<tr>
<td>III.1.2.3 Output</td>
<td>34</td>
</tr>
<tr>
<td>III.2 Treatment</td>
<td>37</td>
</tr>
<tr>
<td>III.2.1 myfiltermap</td>
<td>38</td>
</tr>
<tr>
<td>III.2.1.1 Purpose</td>
<td>38</td>
</tr>
<tr>
<td>III.2.1.2 Code description</td>
<td>39</td>
</tr>
<tr>
<td>III.2.1.3 Output</td>
<td>42</td>
</tr>
<tr>
<td>III.2.2 Post-fusion treatment: myfilterfused</td>
<td>43</td>
</tr>
<tr>
<td>III.2.2.1 Purpose</td>
<td>43</td>
</tr>
<tr>
<td>III.2.2.2 Code description</td>
<td>44</td>
</tr>
<tr>
<td>III.2.2.3 Output</td>
<td>44</td>
</tr>
<tr>
<td>III.3 Data fusion</td>
<td>45</td>
</tr>
</tbody>
</table>
III.3.1 Purpose ................................................. 45
III.3.2 Code description ....................................... 45
III.3.3 Output .................................................. 47
III.4 The Graphic User Interface ............................... 49
III.4.1 Purpose .................................................. 49
III.4.2 Description .............................................. 49
III.4.3 Integration within CEASIOM ......................... 56
IV Conclusion .................................................. 57

Appendices .................................................. 60
I Kungliga Tekniska Högskolan (KTH) - The Royal Institute of Technology[1] .... 60
  I.1 General points ............................................ 60
  I.2 Key numbers .............................................. 60
  I.3 Miscellaneous ............................................ 61
  I.4 The Aerodynamics division .............................. 61
    I.4.2 Activities[5] .......................................... 62
    I.4.3 Organisation .......................................... 63
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Moments on an aircraft</td>
</tr>
<tr>
<td>2</td>
<td>Anatomy of an airplane</td>
</tr>
<tr>
<td>3</td>
<td>Influence of the control surfaces</td>
</tr>
<tr>
<td>4</td>
<td>Angle of attack $\alpha$</td>
</tr>
<tr>
<td>5</td>
<td>Sideslip angle $\beta$</td>
</tr>
<tr>
<td>6</td>
<td>Sideslip angle $\beta$</td>
</tr>
<tr>
<td>7</td>
<td>CFD problem solving approach</td>
</tr>
<tr>
<td>8</td>
<td>Map of the partners in the SimSAC project</td>
</tr>
<tr>
<td>9</td>
<td>The CEASIOM modules</td>
</tr>
<tr>
<td>10</td>
<td>Format of the aerodynamic tables regarding respectively the state and the control parameters.</td>
</tr>
<tr>
<td>11</td>
<td>The GUIDE layout editor.</td>
</tr>
<tr>
<td>12</td>
<td>Example of output for the myplot function: the drag coefficient $C_D$ for the $[\alpha$ Mach $\beta]$ site for three original data sets.</td>
</tr>
<tr>
<td>13</td>
<td>Detail of an example of output for the myplot function.</td>
</tr>
<tr>
<td>14</td>
<td>Example of output for the myplot function for treated and fused data.</td>
</tr>
<tr>
<td>15</td>
<td>Illustration of incompatible data sets spotted using myplot.</td>
</tr>
<tr>
<td>16</td>
<td>Illustration of an erroneous data set spotted using myplot.</td>
</tr>
<tr>
<td>17</td>
<td>Illustration of one of the output figures for myvisual.</td>
</tr>
<tr>
<td>18</td>
<td>Illustration of outliers and suspicious points possibly coherent with the pattern of the data.</td>
</tr>
<tr>
<td>19</td>
<td>Illustration of the purpose of the mapping process</td>
</tr>
<tr>
<td>20</td>
<td>Illustration of the effect of the mapping.</td>
</tr>
<tr>
<td>21</td>
<td>Illustration of the architecture of a three-parameter site aerodynamic table.</td>
</tr>
<tr>
<td>22</td>
<td>Tables obtained during the mapping for an $[\alpha$ Mach $\beta]$ site - $C_x$=$f(\alpha)$</td>
</tr>
<tr>
<td>23</td>
<td>Tables obtained during the mapping for an $[\alpha$ Mach $\beta]$ site - $C_x$=$f($Mach number$)$</td>
</tr>
<tr>
<td>24</td>
<td>Tables obtained during the mapping for an $[\alpha$ Mach $\beta]$ site - $C_x$=$f(\beta)$</td>
</tr>
<tr>
<td>25</td>
<td>Illustration of the effect of the mapping.</td>
</tr>
<tr>
<td>26</td>
<td>Illustration of the format of a fused aerodynamic Table.</td>
</tr>
<tr>
<td>27</td>
<td>Illustration of the &quot;multiple site&quot; phenomenon in a fused table.</td>
</tr>
<tr>
<td>28</td>
<td>$C_{roll}$ for $[\alpha$ Mach $\beta]$ site, treated and fused data compared.</td>
</tr>
<tr>
<td>29</td>
<td>Main window.</td>
</tr>
<tr>
<td>30</td>
<td>Dialog box for the selection of data set files.</td>
</tr>
<tr>
<td>31</td>
<td>Message box for the exporting of several arrays to the Matlab Workspace.</td>
</tr>
<tr>
<td>32</td>
<td>About message box.</td>
</tr>
<tr>
<td>33</td>
<td>Help message box.</td>
</tr>
<tr>
<td>34</td>
<td>Warning message appearing when no site to plot is selected and &quot;Plot&quot; is clicked on.</td>
</tr>
<tr>
<td>35</td>
<td>Warning message appearing when no data to plot is selected and &quot;Plot&quot; is clicked on.</td>
</tr>
<tr>
<td>36</td>
<td>Output of the &quot;help Fusion&quot; command.</td>
</tr>
<tr>
<td>37</td>
<td>Organization chart of the CFD division within the KTH Aeronautical and Vehicle Engineering department.</td>
</tr>
</tbody>
</table>
Introduction

The use of Computational Fluid Dynamic in the aircraft industry has been increasing significantly for the past few years, partly because of the increase in reliability of its techniques, partly because it brings the advantage to reduce significantly the production costs. While the previous approach would lead to significant errors being noticed only at wind-tunnel test stages, where costly prototypes would already have been built, thanks to CFD, errors or flaws in the design can be spotted early on in the process, and rapidly corrected or improved upon. In this regard, the industry demands more and more reliable CFD tools.

This is the goal of the SimSAC project. The SimSAC project is an FP6 European project. Coordinated by Pr Arthur Rizzi from the KTH Aeronautical and Vehicle Engineering department, it counts 17 partners, spread all over Europe, of which several key companies and research centers in the world of aeronautics, such as SAAB, Dassault or EADS.

The result is CEASIOM, acronym for Computerized Environment for Aircraft Synthesis and Integrated Optimization Method. It is made of six different modules: AcBuilder, AMB, SDSA, NeoCASS, FCSDT and DSS, that each fulfill a particular mission in the design and characteristics prediction process.

To analyze the performances of a given design, CEASIOM builds aerodynamic tables. These tables contain the state and control parameters and the resulting values of key aerodynamic coefficients, such as the lift, drag and pitch moment coefficients for this specific design.

These tables are built using adaptive fidelity methods: CFD has several techniques at its disposal; for simple flight conditions, simple methods will be used to predict the aerodynamic coefficients, while for complex flight conditions, including turbulences for instance, more advanced methods will be used.

This is done to improve the computational costs: in simple flight conditions, the simple methods will give good results, and there is no need for advanced methods, that require important computational costs. For complex flight conditions, the simple methods are no longer valid. For instance, they almost never include the apparition of turbulences. In these conditions, the use of advanced CFD method is needed.

As a result, the engineer obtains several aerodynamic tables, each covering a domain of the flight envelope. These tables need to be fused in order to obtain a single coherent aerodynamic table covering the whole flight domain, that can be used in the prediction of the characteristics of the aircraft.

The mission was to implement an automated fusion process for any number of different data sets, and to integrate it as a new CEASIOM module.

This module would contain several additional features: a filter, to get rid of aberrant points, a function realizing an automated custom-made mapping of the values, as well as visualization tools.

The whole was to be integrated as one of the CEASIOM modules, implying the development of a Graphic User Interface of its own, as well as producing outputs usable by the SDSA module.

This paper presents the elaboration of the Fusion module and its results.
I Presentation

I.1 Kungliga Tekniska Högskolan (KTH) - The Royal Institute of Technology[1]

This mission was taking place within the Aerodynamics division of the Kungliga Tekniska Högskolan (KTH) department of Aeronautical and Vehicle Engineering.

Located in Stockholm, KTH, or the Royal Institute of Technology, was founded in 1827. It includes courses and research programs in various fields, from all the branches of engineering, to natural sciences, architecture, urban planning or management. It also houses several competence centers, and research programs financed by various scientific foundations. It is also part of a vast network of partnerships with other universities, in Sweden as well as in all parts of the world: Europe, Australia, America and Asia.

KTH Aeronautical and Vehicle Engineering department is part of the School for engineering sciences[4]. It concerns itself with the science and research behind air, rail, ground and sea vehicles.

The researches carried out in this division focus on the study of the present aerodynamic limitations, as well as to investigate new aeronautical concepts, by using, enhancing or creating computational tools, hence applied or experimental CFD.

It possesses extensive experimental resources, such as a L2000 wind tunnel, in order to validate the theoretical results obtained by the researchers.

The possible applications are vast, and include, for instance, providing computer-based design studies, thus limiting the need for costly experiments.

The department has a vast network of partners at national and international level, including government agencies and academic institutions.

The division has been involved in several European funded projects[5]:

- the Rear fuselage and EMPennage Flow Investigation (REMFI) project, that investigated the complex physics of the flow in order to come up with innovative and more efficient tail designs,
- the HISAC project, aiming at finding an environmentally friendlier high speed aircraft, that involved 37 partners in 13 different countries (ended end of 2009),
- the VFE-2,
- the SimSac project, that will be developed later in this paper.

The department is also involved in the development of two softwares: Tornado and LIC.

- LIC, for Line Integral Convolution is a method providing a visualization of the flow patterns in a similar way as the ones that can be observed in wind tunnel testing in wind-blown sand or oil-flow procedures.
- Tornado is a wing design code developed within a collaboration between KTH, the University of Bristol, the Linköping University and the Redhammer Consulting Ltd. It allows the user to obtain quasi-immediate feedback on changes in design in terms of the aircraft’s performances, thus providing the engineers with useful information early in the design process. Constantly developed and updated, Tornado is used in universities and companies around the world and has become a prominent software in the world of aircraft design.

The department currently has responsibility for the VINNEX Centre of Excellence for ECO2 Vehicle design and coordinates the Gröna Taget research program.

The researchers in the department have also published several papers and books of scientific values in the past years.\(^1\)

\(^1\)For more information on the Royal Institute of Technology see Appendix 1.
I.2 Basic Aerodynamics notions

In order to get a better understanding of the mission, a few notions have to be introduced.

I.2.1 Aerodynamic Forces and Moments[9]

During flight, the air around the aircraft changes from the one in the undisturbed stream. Its speed changes due to the deflection it is submitted to; Bernouilli showed that the pressure it exerts on the plane is also different. Finally, friction forces appear due to the viscosity of the air.

These processes create a resultant force and moment. By convention, they are each separated into 3 components.

The resultant force is the sum of:

- the lift $L$ : as its name indicates, it is the force acting upwards.\footnote{The word ‘upwards’ is used in the same sense than the pilot’s head is above his feet, not in the predicate than the ground represents the most downward direction. This is the sense applied to the word ‘upwards’ in the integrality of this paper. By extension, the associate opposite, ‘downward’, is defined as the opposite of ‘upwards’ as defined here.} It is perpendicular to the direction of the flight or of the undisturbed stream.

- the drag $D$ : in the adverse direction to the line of flight, it is the force resisting the motion.

- the cross-wind force $Y$ : it is the force perpendicular to both lift and drag. It is considered positive when acting toward the right-hand of the pilot.

The resultant moment is the sum of:

- the pitching moment $(M)$ : it creates a rotation of the aircraft about the axis perpendicular to the longitudinal plane of symmetry of the plane. It is considered positive when raising the nose of the plane upwards.

- the rolling moment $(LR)$ : it creates a rotation of the aircraft about the longitudinal axis of the plane. It is considered positive when acting to depress the starboard wing-tip.

- the yawing moment $(N)$ : it creates a rotation of the aircraft about the vertical body axis of the plane. It is considered positive when swinging the nose of the plane to the right.

![Figure 1: Moments on an aircraft](image)
All these forces and moments are characterized by non-dimensional coefficients: $C_L$, $C_D$, $C_Y$, $C_m$, $C_R$, $C_N$ respectively for the lift, the drag, the cross-wind force, the pitching moment, the rolling moment and the yawing moment.

With $F$ being one of the 3 components of the resultant force, and $Q$ one of the 3 components of the resultant moment, the aerodynamic coefficients are defined as follow:

\[ C_F = \frac{F}{1/2\rho V^2 S} \]  
\[ C_Q = \frac{Q}{1/2\rho V^2 Sl} \]

where $l$ is a reference length and $S$, in the case of a fuselage, is a surface defined by the engineer, and can be the projected formal area, the maximum cross-sectional area or the $(volume)^{(2/3)}$.

I.2.2 Anatomy of an aircraft[2]

The schematic of a typical aircraft is presented in the figure below. It indicates the commonly known parts, such as the fuselage, the wings and the gears, as well as the more specific ones, such as the stabilizers and what are called the control surfaces: the ailerons, the rudder, the elevators and the optional trim tab and flaps.

\[ \rightarrow \text{The horizontal stabilizer, or horizontal tail resists the pitch moment. It creates a force called downforce, that balances the lift produced by the wing, thus enhancing the global stability of the aircraft.} \]
The vertical stabilizer acts in the same way, but related to the yaw moment. It thus produces a sideforce that will counteract a side motion of the aircraft.

The control surfaces allow the pilot an enhanced control of the aircraft, by acting on the aerodynamic forces and moments.

→ The ailerons, located at the end of each wing, acts on the lift. They work in opposite directions: when one is deflected of a given angle upward, the other is deflected of the same angle downward. Deflecting the aileron of a wing downward increases the lift, while deflecting it upward decreases the lift, thus the wing with the aileron down will roll upward. If the aileron on the starboard wing is down, the plane will roll to the left and vice-versa. The angle of deflection of the ailerons is considered positive downwards.

→ The elevators, located at the rear of each wings of the horizontal stabilizer, acts on the downforce, and thus on the global pitch moment. When deflected upwards, the downforce increases, the nose of the plane then pitches upwards. The angle of deflection is positive in this configuration.

→ The rudder, located at the rear of the vertical stabilizer acts on the side force produced by the vertical stabilizer and thus on the global yaw moment. When deflected to starboard, the side force on the outboard increases, the nose of the plane yaws to starboard. The angle of deflection is positive in this configuration.

The effects of the main control surfaces can be summarized by the figure below.

![Figure 3: Influence of the control surfaces](image-url)

The trim tab and flaps are optional control surfaces.

→ The trim tab can be located on all control surfaces, though it is mostly used in conjunction with elevators. Its purpose is to maintain the control surface in its equilibrium position. The trim tab position is set beforehand by the pilot, who then avoids the fatigue of constantly manually adjusting the elevator back to its equilibrium position.

→ The flaps are usually located on the wings, between the fuselage and the ailerons. Their function is to increase the lift created by both wings at a given speed. To that effect, they can only move coordinately downwards and are mostly used during key phases of the flight such as take off or landing, that are then possible at lower speeds.
I.3 Parameters of importance

To characterize the state of an airplane, several parameters are considered. However, in this mission, three of them are consistently used.

→ The angle of attack, commonly noted $\alpha$, is the angle formed by the longitudinal axis and the projection of the speed on the $xz$ plane, as shown in the figure [11] below.

![Figure 4: Angle of attack $\alpha$](image)

$\rightarrow$ The Mach number is a dimensionless number characterizing the speed of an object moving through a fluid. It is obtained with the following formula :

$$M = \frac{V}{a}$$

where :

$\quad$ – $V$ is the speed of the object through the fluid
$\quad$ – $a$ is the speed of sound in the fluid for the same temperature and pressure conditions.

In the case of an aircraft, $V$ is the speed of the aircraft and $a$ the speed of sound in air for the same temperature and pressure conditions.

→ The sideslip angle is the angle formed by the longitudinal axis of the plane and the projection of the speed on the $xy$ plane, as shown in the figure [11] below.

![Figure 5: Sideslip angle $\beta$](image)
I.4 Computational Fluid Dynamics

I.4.1 General points

Computational Fluid Dynamics is a branch of fluid mechanics that uses numerical methods based on mathematical representations to analyze, simulate and solve flow problems around bodies defined by boundary conditions. CFD is thus at the crossbow of several connected fields: numerical analysis, computer sciences, mathematics, and, in our case, aerodynamics. The root of all CFD problems can be expressed by the Navier-Stockes equations:

\[
\frac{\partial \rho}{\partial t} + (u \cdot \nabla) \rho + \rho \nabla \cdot u = 0 \tag{4}
\]

\[
\frac{\partial u}{\partial t} + (u \cdot \nabla) u + \frac{1}{\rho} \nabla p = \frac{\mu}{\rho} \nabla^2 u - \frac{1}{3} \nabla (\nabla \cdot u) \tag{5}
\]

\[
\frac{\partial p}{\partial t} + (u \cdot \nabla) p + \gamma p \nabla \cdot u = (\gamma - 1) \left[ (\tau \cdot \nabla) u + \nabla \cdot (k \nabla T) \right] \tag{6}
\]

Where:

- \( \rho \): the field of density
- \( u = (u, v, w)^T \): the Cartesian velocity vector
- \( p \): the pressure
- \( \gamma \): the ratio of specific heat
- \( \tau \): the shear stress tensor, which for a Newtonian fluid is:
  \[
  \tau = \mu \left[ \nabla u + (\nabla u)^T \right] - \frac{2}{3} \mu (\nabla \cdot u) I
  \]
- \( T \): the temperature
- \( k \): constant

These equations are highly complex, and have been successively simplified for resolution in the early years of the CFD.

By neglecting the viscosity terms, one obtains what is called the Euler equations, characterizing inviscid flows:

\[
\frac{\partial \rho}{\partial t} + (u \cdot \nabla) \rho + \rho \nabla \cdot u = 0 \tag{7}
\]

\[
\frac{\partial u}{\partial t} + (u \cdot \nabla) u + \frac{1}{\rho} \nabla p = 0 \tag{8}
\]

\[
\frac{\partial p}{\partial t} + (u \cdot \nabla) p + \rho a^2 \nabla \cdot u = 0 \tag{9}
\]

Where \( a^2 = \gamma \left( \frac{E}{\rho} \right) \) is the square of the speed of sound in air.

By removing the vorticity term, one obtains what is called the Full Potential equations, characterizing irrotational compressible and inviscid flows:

\[
(1 - M^2_x) \frac{\partial^2 \Phi}{\partial x^2} + (1 - M^2_y) \frac{\partial^2 \Phi}{\partial y^2} + (1 - M^2_z) \frac{\partial^2 \Phi}{\partial z^2} - 2M_x M_y \frac{\partial^2 \Phi}{\partial x \partial y} - 2M_y M_z \frac{\partial^2 \Phi}{\partial y \partial z} - 2M_z M_x \frac{\partial^2 \Phi}{\partial z \partial x} = 0 \tag{10}
\]

With:

\[
M_x = \frac{1}{a} \frac{\partial \Phi}{\partial x}
\]
\[ M_y = \frac{1}{a} \frac{\partial \Phi}{\partial y} \]

\[ M_z = \frac{1}{a} \frac{\partial \Phi}{\partial z} \]

Where \( a \) is the local speed of sound and \( \Phi \) the velocity potential, with the flow velocity \( \mathbf{v} = \nabla \Phi \).

And finally, by linearizing the velocity potential into an undisturbed velocity \( V_\infty \) in the \( x \) direction and a small perturbation velocity \( \nabla \varphi \):

\[ \nabla \Phi = V_\infty x + \nabla \varphi \]  

(11)

The Linearized Potential Equations is obtained:

\[ (1 - M_\infty^2) \frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} + \frac{\partial^2 \varphi}{\partial z^2} = 0 \]  

(12)

The first equations to be solved were thus the linearized potential equations, for the flow around an airfoil, first with a 2D method in 1930, and then with 3D methods in 1967 with the progresses in computer sciences. Additional features to this method, taking lifting panels into account, were developed and described in a paper by Paul Rubbert and Gary Saaris of Boeing Aircraft in 1968.

The first codes for airfoils analysis and design appeared in the early 1980s, with codes like PROFIL or Xfoil by Mark Drela (MIT), that is still developed and used nowadays.

The Full Potential Equations were solved in order to bring more accuracy in the calculations, since the so-called panel codes used so far could not take into account the non-linear flow appearing at transonic speeds. The first description of such a code was given by Earll Murman and Julian Cole of Boeing in 1970.

The next step was solving the Euler equations, which brought even higher accuracy for transonic flows. Codes such as MSES, developed in 1986 by Mark Drela and Michael Giles (MIT), appeared, and have been further developed to be used in the analysis, design and optimize single or multi-element airfoils. MSES is still widely used nowadays.

Finally, the Navier-Stokes equations, referred to as RANS or URANS, were solved, with first the Nasa Ames’ ARC2D code, and are now commonly used in different codes and programs for flow calculations requiring high level of accuracy.[7]

The figure below illustrates the accuracy of the different methods.
The values obtained with the Vortex-Lattice Method (VLM), Euler and RANS are compared to experimental (wind-tunnel) data. The Vortex-Lattice method is the most simplified. The results are quite accurate at the beginning of the curve, but Euler gives even better results, while RANS is almost entirely at one with the experimental values on the tested domain.

I.4.2 Methodology

All CFD problem solving approaches follow the same steps, categorized into three main phases, as shown in the diagram below.

During the pre-processing stage, the main flow phenomena involved in the problem is identified, and its mathematical representation constructed. This involves analyzing partial differential equations and defining boundary conditions. The problem is then numerically formulated: the geometry of the problem is defined, an appropriate mesh of the volume occupied by the fluid is constructed, and the differential equations are implemented, as well as the initial and boundary conditions. During the running stage, the simulation is started and the solution obtained. The obtained results are analyzed and possibly visualized during the post-processing phase in order to be interpreted in order to draw the final conclusions. Finally, the results are compared to wind-tunnel data for validation.
I.4.3 Applications

The CFD applications are mostly of the design domain, which touches different aspects of the construction of a plane. First, its shape: on the outside, CFD has provided the engineers with several accurate «design & analyze» tools, that are used to optimize the shape of the airfoils, the wings or the empennage, for better performance. On the inside, CFD plays a role in constructing an optimal integration of the engine or weapons, by providing informations on the influence of using devices such as pylons, nacelles, inlets, diffusers or nozzles.

Another application is in the study of the mere performances of the plane. The complex calculations at the base of CFD give the engineers access to the performances of the plane in terms of lift, drag and moments. It is also possible to determine the stability & control and handling characteristics. The structural and aeroservoelastic designs also benefit from CFD. By computing the surface pressures, the load can be determined and the influence of aeroelastic impacts on the performances calculated, while also taking into account the effects of the control system.
I.5 The SimSAC project

I.5.1 Presentation[3]

The SimSAC project, for Simulating Aircraft Stability and Control CHaracteristics for Use in Conceptual Design, is an FP6 european project. Coordinated by Professor Arthur Rizzi from KTH, it counts 17 partners, spread all over Europe, of which several key companies in the world of aeronautics:

- The Royal Institute of Tecnology (KTH)
- The Swedish Defence Research Agency (FOI)
- SAAB Group Saab Aerosystems
- Alenia Aeronautica
- The Politecnico di Milano
- The University of Bristol (BRI)
- The University of Liverpool (LIV)
- The European Center for Research and Advanced Training in Scientific Computing (CERFACS)
- The European Aeronautic Defence and Space Military Aircraft Business Unit (EADS-M)
- Dassault Aviation
- Computational Fluids & Structures Engineering (CFS)
- The German Aerospace Centre (DLR)
- J2 Aircraft Dynamics
- Onera
- The Central Hydrodynamic Institute (TsAGI)
- The Aeronautical and Research Institute (VZLU)
- The Warsaw University of Technology

![Figure 8: Map of the partners in the SimSAC project](image)
The SimSAC project focuses on the elaboration and amelioration of the current CFD tools in the calculation of the dynamic stability & control differential coefficients, and their integration in a new software for conceptual design and analysis of fixed-wing aircrafts. Called CEASIOM (Computerized Environment for Aircraft Synthesis and Integrated Optimisation Methods), it has so far been applied to design studies on concept - such as a TCR (near-sonic large transport aircraft) and a Z-wing general aviation configuration - as well as existing airplanes: the Ranger 2000 military trainer, and a supersonic jet.

I.5.2 Motivation

An efficient aircraft design tends to improve the stability of the aircraft and expand its flight envelope. This requires an accurate description of the plane at all states, linear and non-linear, in order to properly design the flight control system.

So far, to help them in their work, designers only had data from previous experiences and/or semi-empirical data. This simple approach gives fine results in the design of basic parameters such as the shape or area of the wings, but was often a source of mistakes when more precise parameters were needed, such as the design and position of the control surfaces, where complex phenomena have to be taken into account, such as the effects of the Reynolds number or of the dynamic motion. These errors would most of the time be detected in the final testing stages, during wind-tunnel testing or even flight tests. Obviously, the later an error is spotted, the more expensive it is to fix, thus the growing interest in producing high-fidelity data, that could guide the designers at all stages of the design, avoiding costly mistakes and ensuring better performances.

I.5.3 CEASIOM[12]

CEASIOM is acronym for Computerized Environment for Aircraft Synthesis and Integrated Optimization Method. Developed within the SimSAC research project, its aim is to provide engineers with reliable predictions of stability and control properties of a given aircraft design at an early stage of the design process. It englobes a set of six different modules: AcBuilder, AMB, SDSA, NeoCASS, FCSDT and DSS, as shown in the figure below.

These modules allow the user to customize the geometry of the airplane, perform aerodynamics, aeroelastics and stability & control calculations, as well as early flight control formulation and provide decision support.

→ AcBuilder, for Aircraft Builder, uses a customized system of surface and volume grid generators to create a visual feedback of the airplane geometry as well as the associated data used for weights and balance estimates.
→ AMB-CFD: performs the aerodynamics calculations using adaptable-fidelity modules referred to as:
  - tier I: steady and unsteady TORNADO vortex-lattice code (VLM) for low-speed aerodynamics and aeroelasticity.
  - tier I+: Inviscid Edge CFD code for high-speed aerodynamics and aeroelasticity.
  - tier II: RANS (Reynolds Averaged Navier-Stokes) flow simulator for high-fidelity analysis of extreme flight conditions.

→ the SDSA module performs the stability and control calculations. It is able to carry out six degree of freedom test flight simulations and to predict the performances of the airplane, with the included influence of a human pilot, a Stability Augmentation System (SAS) or a Flight Control System.

→ NeoCASS produces the calculations involving aeroelasticity problems.

→ FCSDT is the module allowing the designing of a Flight Control System.

→ DSS is the Decision Support System module.

CEASIOM works as a complete design tool, within the frame of a Design Simulate and Evaluate (DSE) exercise. With design specifications, the engineer is able to begin with a baseline configuration. All calculations needed, as well as visual feedback, are completed within the AcBuilder and AMD modules. The result is an aerodynamic table characterizing the forces and moments applied to the aircraft. These tables are then analyzed by the SDSA module to obtain knowledge on the flying qualities of the studied aircraft. The baseline can then be improved upon on account of these informations, in the design as well as with possible addition of appropriate SAS or FCS, whose specifications can be predicted with the FCSDT and DSS modules. The modifications can then be studied in the same way than the baseline configuration and improved upon if needed.
II Objectives & Mission

II.1 Presentation and Motivations

To produce an accurate prediction of the Stability & Control characteristics of an aircraft, the SDSA module needs a complete aero-database. Produced by the AMB-CFD module, it comes in the form of a table of dimensionless aerodynamic coefficients characterizing the forces and moments the aircraft is subjected to. These coefficients are given as a function of the state vector and control-surface deflection angles. The state vector is \([\alpha \ M \ \beta \ q \ p \ r]\), where:

- \(\alpha\) is the angle of attack.
- \(M\) is the Mach number.
- \(\beta\) is the sideslip angle.
- \(q, p \text{ and } r\) indicate the respective rotation angles in pitch, roll and yaw.

The elevators, rudder and ailerons are the only control surfaces considered. The control vector is thus \([\delta_e \ \delta_r \ \delta_a]\), indicating the respective deflecting angle for the elevators, the rudder and the ailerons.\(^3\).

The tables have the following format:

<table>
<thead>
<tr>
<th>(\alpha)</th>
<th>(M)</th>
<th>(\alpha)</th>
<th>(\beta)</th>
<th>(q)</th>
<th>(p)</th>
<th>(r)</th>
<th>(C_L)</th>
<th>(C_D)</th>
<th>(C_m)</th>
<th>(C_Y)</th>
<th>(C_{roll})</th>
<th>(C_n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>-</td>
<td>x</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

*Figure 10: Format of the aerodynamic tables regarding respectively the state and the control parameters.*

These tables provide reliable data. However, the problem lies in the computational cost: with one calculation for every entry in the table, to obtain a substantial enough database would require more than thousands of calculations. Calculations that, depending on their level of fidelity and complexity, require by themselves a certain amount of time.

Running advanced computational methods such as RANS on the whole domain of the flight envelope would prove extremely expensive and time-costly, while not necessary. Indeed, for early design, and for common flight conditions, “simple” tier 1 or tier 1 + methods demand less in terms of computational costs and are sufficient to provide accurate data.

The high-fidelity methods are only truly needed in advanced considerations, for instance when the flight envelope conditions are reached and that non linear effects, turbulences for instance, appear. A solution to reduce the computational costs is to proceed in this manner: using low-fidelity CFD methods in early design and in common flight conditions, and advanced, more time and money consuming, methods in complex flight conditions.

That is the solution implemented in the AMB-CFD CEASIOM module. As a result, the engineer obtains several tables with different fidelity level, each one covering a part of the flight domain. In order to obtain a single coherent data base on the whole flight domain, these tables need to be combined. Because of the large dimensions of these tables, the task would be manually very long and error prone.

To automatize this process would greatly reduce the amount of time needed as well as possible mistakes, thus improving the overall reliability. That is the task I completed during my master’s thesis.

In addition to the fusion process, several parameters had to be taken into account.

As part of any such software, outlier points can appear and alter the resulting table. These points needed to be filtered out before the fusion process.

\(^3\)with the convention indicated in section II.1.2
Moreover, the repartition of the values for some coefficients is such that jumps in the values can appear. They had to be taken into account so that the fusing process would remain accurate. To that effect, the domain had to be custom-mapped automatically.

Finally, an interesting addition was to provide the engineer with a visual of the data.

The whole was to be integrated as one of the CEASIM modules, implying the development of a Graphic User Interface of its own, as well as producing outputs usable by the SDSA module.

In this respect, the common format in the CEASIM software has to be presented. The tables that are of interest in the developing of the Fusion module are only a part of the data sets. These data sets are produced by the AMB-CFD module and saved as .xml format. They contain all the information relevant to the calculations carried out in the SDSA module, and coming from all the previous modules. This include information on the pilot, the geometry of the plane, its motor characteristics and much more. All these informations are stored in the .xml file following a precise outline. This outline must be kept, partly because all the information contained in it are essential and must be conveyed to SDSA after the Fusion, partly because the SDSA module will only handle files formatted according to this outline.

II.2 Scope of work

Bearing all these considerations in mind, the scope statement can be expressed by the following requirements:

→ Implementing a function carrying automatically the fusion of any number of data sets, with an output compatible with the SDSA module.

→ Implementing a function automatically spotting the outliers and filtering them out.

→ Implementing a function automatically detecting whether a custom-made mapping is needed, and creating the appropriate one.

→ Implementing a visualization function for all the data.

→ Realizing the integration of these functions into a Matlab GUI

→ Integrating the GUI into CEASIM’s GUI.

II.3 Principle: kriging — an interpolation process.

In order to fuse the data sets into a single coherent one, the missing data needs to be interpolated from the known data. Kriging — named after the south african engineer Daniel G. Krige, upon whose work French mathematician Georges Matheron developed it — is a group of statistical techniques from the family of linear least squares estimation algorithms. It interpolates the value of a function at a point \( x \) given the values obtained for points \( x_1, x_2, ..., x_n \) at adjacent locations.[6]

In our case, the values to interpolate were the aerodynamic coefficients; the "x points", the missing state & control vectors; and "\( x_1, x_2, ..., x_n \) at adjacent locations", the tested state & control vectors, for which the aerodynamic coefficients are known, that are close in terms of distance to the missing state and control vectors the aerodynamic coefficient is being interpolated for.
II.4 Tools

Matlab, the well-known programming environment, was chosen for this mission. Created in the late 1970’s by Cleve Moler, then chairman at the computer science department at the University of New-Mexico, Matlab is now developed by Mathworks, company he founded with an engineer named Jack Little. It is nowadays widely used in mathematics and physics oriented calculations, in the industrial as well as academic environments. It was thus the best option for our project. The fact that the CEASIOM modules were all coded in Matlab was another strong argument in favor of its use for this mission. As part of the development, several Matlab toolboxes were used.

II.4.1 The DACE Toolbox[10]

The DACE Toolbox was developed by Søren N. Lophaven, Hans Bruun Nielsen and Jacob Søndergaard, and is titled by its developers "A Matlab Kriging Toolbox". It presents several advantages for our purpose:

- it is proposed free and open source, which can be useful in case modifications are needed;
- it is implemented in Matlab code;
- it proposes a complete kriging package, with regressions and correlation functions;
- it was written for high dimensionnal inputs, which is exactly our situation — state vectors count 6 parameters, control vectors 5, which gives an input of a total of 11 parameters—;
- it uses deterministic processes, meaning the same results are obtained from the same points at every run. This is important in terms of reliability of computations.

The functions used for the fusion are presented below.

→ the **dacefit** function

```
[dmodel] = dacefit(S,Y,regr,corr,theta0,lob,upb)
```

It builds a model ("dmodel") from the given set of data, regression and correlation functions.

- S and Y contain respectively the sites— the "x" points —, and their corresponding values— the "f(x)" values —.
- *regr* and *corr* are handles to the regression and correlation functions chosen to interpolate the values at untried sites.
  The package includes zero, first and second order polynomial regression functions, as well as the most common correlation functions: exponential, generalized exponential, Gaussian, linear, spherical and cubic spline.
- *theta0* is the initial correlation length, *lob* and *upb* are the lower and upper boundary of the correlation length.

→ the **predictor** function

```
y = predictor(x,dmodel)
```

It uses the model established with dacefit to predict the values at untried sites.

- *x* is an untried site in a row vector
- *dmodel* is the model obtained from dacefit
- *y* is the resulting interpolated value(s).

→ the **dsmerge** function

23
It is an auxiliary function, included in the DACE Toolbox. The dacefit function can not work with "multiple sites", sites that are so close that they are considered the same. The dsmerge function merges these sites into a single one, according to several options specified by the user.

- \( S \) and \( Y \) are respectively the known sites and their corresponding values,
- \( ds \) indicates the threshold for which two sites are considered equal,
- \( nms \) indicates the choice of norm in which the distances between sites are measured. There are two options:
  * 1 uses the sum of absolute coordinate differences.
  * 2 uses the Euclidian distance. It is the default option.
- \( wtds \) is an option indicating what to do with the multiple sites values in \( S \). There are 3 options:
  * 1 returns the mean value. It is the default option.
  * 2 returns the median value.
  * 3 return the cluster level value
- \( wtdy \) is an option indicating what to do with the values in \( Y \) corresponding to multiples sites. There are 5 options, the 3 first being the same as \( wtds \), the 2 last being returning the minimum value and returning the maximum value.

II.4.2 the XML Toolbox

Developed by Marc Molinari of the University of Southampton for the Geodise project, this Matlab toolbox contains several functions for the manipulation and conversion of data to and from .xml files. It is used exclusively at the very beginning and end of the program.

As explained, the Fusion module is used after the data sets have been generated by the AMB module. They are, as outputs of this module, saved as .xml files.

However, to be manipulated by the different functions, the data need to be loaded into Matlab. This is achieved by the \texttt{xml_load} function.

\begin{verbatim}
xml_load(filename)
\end{verbatim}

- \( filename \) is the name of the .xml file to load into Matlab.

The XML toolbox is also used at the end of the program.

The output of the Fusion module, the fused data, is stored as a variable within the GUI and possibly exported to the Matlab workspace.

In order to be used by the next module, SDSA, to carry out the Stability & Control calculations, the fused data needs to be completed with additional information and formatted specifically to then be stored in a .xml file compatible with SDSA, .

The conversion of the newly formatted Fused data to .xml is realized by the \texttt{xml_save} function.

\begin{verbatim}
xml_save(filename,var)
\end{verbatim}

- \( filename \) is the name the user chooses to give the .xml file
- \( var \) is the name of the variable to store in the filename.xml file
II.4.3 GUIDE: Matlab’s tool for GUI creation

For the development of a Matlab integrated GUI for my program, I chose to use GUIDE, the "GUI Design Environment" provided by Matlab.

→ The first step in building a GUI with GUIDE consists in creating the layout of the GUI. This can be done from scratch, or by selecting one of the pre-made layouts proposed in the menu, and possibly modifying them. The layout editor then opens up. It proposes a complete array of buttons and controls for your GUI: push buttons, sliders, radio buttons, check boxes, Edit Text bars, Static Text bars, Pop-up menus, Listbox, Toggle Buttons, Tables, Axes, Panels, Button groups and Activex Controls. A simple drag and drop from the menu bar will position them on the layout, where they can then be resized, aligned, and their properties defined.

![Figure 11: The GUIDE layout editor.](image)

This creates a .fig figure, that, once saved, will generate a .m file containing initializing code, as well as the headers for different callbacks functions associated to the buttons present on the GUI. If the program is run at this point, though the GUI will appear, nothing will happen if you click a button.

→ The second step is then to program the different components of the layout, by writing the callback functions in the .m file. This type of event oriented programming is slightly different than the regular Matlab language. It uses button specific handles to create variables and operate on them.

Once this is done, the GUI is complete and operates as programmed.

II.4.4 Suplabel Toolbox

Developed by Ben Barrowes, this toolbox was used in the visualizing functions to produce clearer, more detailed types of labels, that could be positioned appropriately for the different types of plotting figures created.
III  Results

III.1  Data visualization

Motivation
When the engineer has generated the different data sets using the modules in CEASIOM, a few steps need to be taken before they can be fused. The first one is to check that no error has been made in the parameters, or in the program itself, ensuing data sets that would not be compatible for fusion. A way to do this is to allow the engineer to visualize, prior to any action, the content of the data sets in a meaningful way. This first step is achieved by two functions: myvisual and myplot, each serving a special purpose in this task.

III.1.1  myplot

III.1.1.1  Purpose
The goal of this function was to provide the engineer with a global but relevant view of the data he has generated, both before the fusion, in order to make sure the data sets are correct and compatible with each other, and after the fusion, to visualize simultaneously all the data sets and the resulting fused data, in order to spot any possible anomaly. The function provides the engineer with plots of all six aerodynamic coefficients, separated by data sets and for each state & control vector for better visibility. It allows the engineer to immediately judge whether an error has been made in the selection of the files or in the making of the file itself.

III.1.1.2  Code description
The function is declared as:

\[
\text{myplot(data, opt, varargin)}
\]

where:

- **data** is a structure array containing the different data sets, — this setting is used at the beginning of the process, when the engineer verifies the data sets he has loaded — or a cell array containing the filtered and mapped data sets — used at the end of the process, when the fused data is compared to the data sets used for the fusion—.

- **opt** is a parameter that allows the user to choose which values he wants to see plotted. As explained earlier, the plottings are separated by data sets and state & control vectors. The parameter **opt** can then take 9 values:
  - ’1’: the function plots each coefficient obtained with the different \([\alpha \text{ Mach}]\) sites for all data sets.
  - ’2’: the function plots each coefficient obtained with the different \([\alpha \text{ Mach } \beta]\) sites for all data sets.
  - ’3’: the function plots each coefficient obtained with the different \([\alpha \text{ Mach } q]\) sites for all data sets.
  - ’4’: the function plots each coefficient obtained with the different \([\alpha \text{ Mach } p]\) sites for all data sets.
  - ’5’: the function plots each coefficient obtained with the different \([\alpha \text{ Mach } r]\) sites for all data sets.
  - ’6’: the function plots each coefficient obtained with the different \([\alpha \text{ Mach } \text{ elevator}]\) sites for all data sets.
  - ’7’: the function plots each coefficient obtained with the different \([\alpha \text{ Mach } \text{ rudder}]\) sites for all data sets.
The body of the function is divided into 3 main parts.

→ The first one contains a test on the number of input parameters, to ensure that the user did not give more than 3 parameters, or less than 2. The test also verifies the nature of the input parameters: the second parameter must be a double, the third parameter — if any — must be an array of double, since that is the format the fused data is stored in. If these conditions are not fulfilled, the adapted error message is displayed:
  
  - "Too many/ Too few input parameters"
  - "The second parameter must be a double"
  - "The third parameter must be an array of double"

→ The second part consists in obtaining a few parameters used in the actual plotting, and dividing up the data by site.
  This is done in order to facilitate the actual plotting of the data by aerodynamic coefficient and site. The way to proceed will differ according to the stage of the process.
  Indeed, as explained earlier, two cases can appear:

  - Case 1: the function is run at the very beginning of the process.
    The data used has just been loaded and is in the form of a cell array containing all the data sets.
  
  - Case 2: the function is run after treating — filtering and mapping — the original data.
    The new data is stored into a structure array.

To ensure an appropriate behavior of the program and the plotting of the correct values, the second part begins with a test on the nature of the first input parameter, `data`. A possible error is also treated, displaying, in case of an erroneous class for the data input, the message: "This function is defined for cell and struct arrays only".

The plotting function produces 6 figures by option, one for each aerodynamic coefficient. Each of these figures is divided into \( n+1 \) lines, one line for each data set, and an additional one in case the fused data is also plotted.

Each of the subplots a line of the figure contains presents the values of one aerodynamic coefficient as a function of \( \alpha \) for each value of the Mach number, and in the case of a three parameter site, which is all of them but the first basic \( [\alpha \text{ Mach}] \) one, for each value of the third parameter. Thus, there need to be at least as many columns per line as there are values for the third parameter. For this reason, the total number of values in all the data sets for each third parameter — \( \beta, q, p, r, \text{elev}, \text{rud}, \text{ail} \) — need to be calculated.

**Case 1** The first cell of the array contains the first data set, the second cell contains the second data set and so on. Each data set, and thus each cell in the array, contains, among other information:
the number of values for each parameters: respectively noted Nalpha, NMach, Nbeta, Nq, Np, Nr, Nelev, Nrud and Nail for the number of values taken respectively by \( \alpha \), Mach, q, p, r, elev, rud and ail.

N.B For the control surfaces such as the elevators and the ailerons, the values of the deflection angles are given for the one located on the starboard, the other one necessarily being deflected of the opposite angle.

the tables containing the data.

In order to divide up the data appropriately, simple calculations are carried out. Given the architecture of the tables:

- the values associated to the \([\alpha \text{ Mach}]\) sites are the ones from line 1 to line Nalpha*NMach.
- the values associated to the \([\alpha \text{ Mach } \beta]\) sites are the ones from line Nalpha*NMach*(1+NBeta).
- the values associated to the \([\alpha \text{ Mach q}]\) sites are the ones from line Nalpha*NMach*(1+NBeta)+1 to line (Nalpha*NMach)(1+NBeta+Nq)...and so on till the end of the table.

The same is done for the controls table.

Case 2 In this case, following the filtering and mapping function, the input contains 10 lines in 1 column, one for each site, one indicating the number of data sets that have been fused, and a last one indicating the size of each site for each data set after the treatment. This is done because simply using the \(N_x\) values as was done in case 1 to calculate the size of each site sample can no longer be done, since points might have been deleted during the treatment process. The 8 sites are themselves divided into 6 cells, for each of the respective aerodynamic coefficients. The "Size" cell is divided into n cells, for each of the data sets, that are themselves divided in 8, for the sizes of the different sites. Using the content of the "Size" cell, the data is divided up into respective sites for each data set.

In both cases, the result is two new cell arrays, S and Y. The S cell array contains the values of the sites, while Y contains the aerodynamic coefficients. S and Y are both divided into n cells, n being the number of data sets. These n cells are each divided into 8 cells, containing each of the 8 sites.

The number of values for each third parameter is also calculated in both cases. One could think that, in case 1, a possible solution would be to take the maximum \(N_x\), but that would be forgetting that the values of that third parameter might all be different from the ones in the other data sets.

As an example, take 3 data sets. The values for q in the first one are [-10 5 10], the values for q in the second one are [-5 5], and finally the values for q in the last data set are [-3 3]. In order to have a working subplot, the fake good idea would be to pick \(m=3\), since this is the highest number of values for q in the different data sets, but if the fused data is plotted, \(m\) must equals 6. Indeed, the fused data has all the values for all the parameters contained in the data sets put together. In this example, this mean that the values for q in the fused data set are [-10 -5 -3 3 5 10], and there need to be a subplot for each of these values.

Finally, the functions being written to work for any number of data sets, all these calculations have to be carried out within a loop on the number of data sets, recuperated at the beginning of the function in the form of the number of cell arrays in case 1, or directly

\[^4\text{The reasons behind this format are explained later on, in the elaboration of the filtermap function.}\]
from the parameter N indicating the number of data sets in case 2.
Once this is done, the third part can begin.

→ The third part consists of the actual plotting.
According to the choice of options, the aerodynamic coefficients are plotted for the chosen site.
The data needs to be separated by values of the Mach number, and values of an eventual third
parameter, for each S and Y arrays.
A test on the number of input parameters indicates to the program whether the fused data must
be plotted alongside the original or treated data sets: if a table is given as a third parameter,
the plotting is carried out for the fused data.

III.1.1.3 Output
Calling the function will render 6 figures, one for each aerodynamic coefficient for the chosen site.

Case 1
The layout of the figures for three original data sets is presented below:

![Figure 12: Example of output for the myplot function: the drag coefficient $C_D$ for the $[\alpha \text{ Mach rud}]$
site for three original data sets.](image)

This figure shows the aerodynamic coefficient $C_D$ for the site $[\alpha \text{ Mach rud}]$.

It is the second of the six figures produced by calling the command:

```matlab
myplot(data, 6)
```

where `data` is a cell array containing the original data sets.
As can be seen, the figure is separated in three lines, one for each data set, and in four columns.
Each figure has a general title, indicating the results of which site are being plotted, here $[\alpha \text{ Mach rud}]$.

Additionally, each subplot also has a title, indicating which data set they represent, and which value
of the third parameter. Below is presented a detail of this figure, in the form of one of the subplots:
The subplot itself presents the aerodynamic coefficient — in this example, $C_D$ — function of $\alpha$, as it is used in this form in aerodynamics, for each value of the Mach number. Each $C_D = f(\alpha, Mach)$ curve is identified by a different color, for which the legend indicates which Mach number it corresponds to.

**Case 2**
The layout of the treated data is the same as for the original data. If the fused data is also plotted, the figure is completed in the n+1 line, as presented below:

**Figure 13:** Detail of an example of output for the myplot function.

**Figure 14:** Example of output for the myplot function for treated and fused data.
III.1.1.4 Examples of use

The figures below illustrate the purpose of this function.

Figure 15: Illustration of incompatible data sets spotted using myplot.

Figure 16 presents the $C_m$ coefficient for the [$\alpha$ Mach q] site, for two data sets. As can be immediately seen from the plots, something is wrong: the first data sets, for all values of $q$, presents a negative slope, while the second data set displays positive slopes for all values of $q$. Since the domain covered by the first data set is also partially covered by the second data sets—in terms of Mach numbers—it is impossible that the result would be so opposed. This means that one, or both the data sets, are wrong. Thanks to this function, the engineer can now spot these problems from the beginning, instead of starting a fusion with incompatible, and almost certainly inaccurate, data sets.

Figure 16: Illustration of an erroneous data set spotted using myplot

Figure 17 presents the $C_L$ coefficient for the [$\alpha$ Mach elev] site, for three data sets. This time, the engineer can immediately see that, in the first data set—first line of plotting—, for the values of the angle of deflection $\delta_e$ of the elevators 5° and 10°, $C_L$ is constant for all values of $\alpha$. 

31
and Mach number.
For the lift coefficient, this is impossible.
The engineer now knows that something went wrong during the computations for these values, and can discard this data set.

Thanks to this function, the engineer can avoid using inappropriate or erroneous data sets from the very beginning of the fusion process, instead of carrying it out, and, in the best case scenario, noticing, once the fusion is done and has given absurd results, that one or several of the data sets were not valid, and in the worst case scenario, not noticing it and carrying on the rest of the calculations.
The engineer also can, once the fusion has given its results, visualize and compare them to the data sets used in the fusion and spot any inaccuracy or erratic behaviour.

III.1.2 myvisual

III.1.2.1 Purpose

A common problem with computer generated data is the apparition of points that are markedly distant from the rest of the data. These points are called "outliers", and it is highly important to filter them out before the fusion, in order to maintain the accuracy of the process and obtain coherent results.
The goal of this function is to identify the outliers.
Since this process must be automatized, a condition had to be defined on exactly how distant from the rest of the data a point had to be in order to be considered an outlier. This condition would then be tested for all points.
In order to spot these points, this condition had to be applied properly: the points should be tested from one data set at a time, for one site at a time, and for one aerodynamic coefficient at a time.
Indeed, the values for $C_D$ in the [$\alpha$ Mach aileron] site in data set 1 might be quite different from its values for the same site in data set 2, and might have nothing to do with the values in $C_Y$, for whichever site or data set.

This condition also had to take into account the specificities of aerodynamic data.
A first idea was to use the slope. Separating the values in a site by Mach number and the third parameter, so that the values would only be a function of $\alpha$, the slope would be tested between the points $x_1$ and $x_2$, then $x_2$ and $x_3$. If the slope was highly different from the previous, the point tested last would be considered an outlier. However, this could not work because of the possible jumps that can be observed in some of the aerodynamic coefficients. The points associated to these jumps, and even probably the points forming the following plateau, would be considered outliers, and subsequently filtered out, ruining the data set, and the future calculations.
It was finally decided to opt for an all-statistical condition. The mean would be calculated for all values of an aerodynamic coefficient — for each site and each data set separately —, and all points far enough of the mean, so as to include a potential jump as valid, would be considered outliers. The "far enough condition" was defined in terms of the standard deviation. The points were then divided up into three categories:

- the "suspicious" points : points that are above or under the mean by two standard deviations.
- the outliers: points that are above or under the mean by three standard deviations or more.
- the valid points, that are none of the above.

The points identified as suspicious or outliers could then be filtered out of the data sets.
However, given the nature of the aerodynamic data sets, where jumps or irregularities can appear without necessarily being uncorrect, the choice was given to the engineer to decide exactly what to do with the identified "suspicious" and "outliers" points. That is one of the reasons why, instead of simply filtering these points out, the myvisual function displays all the points, with special markers on the ones identified as suspicious or outliers, so that the engineer can see how they relate or not to the rest of the data, or judge of the distance between them and the rest of the data.
He can then decide whether to filter out all of them, suspicious and outliers alike, only the outliers,
or — though this is not recommended — to not take any action.

III.1.2.2 Code description
The function is declared as:

[outliers, suspicious, stats] = myvisual(data)

where:

output parameters

- outliers is a cell array. Each cell represents one data set.
  Each of these cells is divided in 8, for the 8 sites: \( \alpha \) Mach, \( \alpha \) Mach \( \beta \), \( \alpha \) Mach \( q \), \( \alpha \) Mach \( p \), \( \alpha \) Mach \( r \), \( \alpha \) Mach elevators, \( \alpha \) Mach rudder, and \( \alpha \) Mach ailerons, respectively named Alpha, Beta, Q, P, R, Elev, Rudder and Ail.
  Each of these sites is divided in 6 for the 6 aerodynamic coefficients, \( C_L \), \( C_D \), \( C_m \), \( C_Y \), \( C_{\text{roll}} \), \( C_n \).
  Each of these cells indicates the position of each outlier this aerodynamic coefficient contains, in this site, in this data set.

- suspicious is a cell array. Each cell represents one data set.
  Each of these cells is divided in 8, for the 8 sites: \( \alpha \) Mach, \( \alpha \) Mach \( \beta \), \( \alpha \) Mach \( q \), \( \alpha \) Mach \( p \), \( \alpha \) Mach \( r \), \( \alpha \) Mach elevators, \( \alpha \) Mach rudder, and \( \alpha \) Mach ailerons, respectively named Alpha, Beta, Q, P, R, Elev, Rudder and Ail.
  Each of these sites is divided in 6 for the 6 aerodynamic coefficients, \( C_L \), \( C_D \), \( C_m \), \( C_Y \), \( C_{\text{roll}} \), \( C_n \).
  Each of these cells indicates the position of each suspicious point this aerodynamic coefficient contains, in this site, in this data set.

- stats is a cell array. Each cell represents one data set.
  Each of these cells is divided in 8, for the 8 sites: \( \alpha \) Mach, \( \alpha \) Mach \( \beta \), \( \alpha \) Mach \( q \), \( \alpha \) Mach \( p \), \( \alpha \) Mach \( r \), \( \alpha \) Mach elevators, \( \alpha \) Mach rudder, and \( \alpha \) Mach ailerons, respectively named Alpha, Beta, Q, P, R, Elev, Rudder and Ail.
  Each of these sites is divided in 6 for the 6 aerodynamic coefficients, \( C_L \), \( C_D \), \( C_m \), \( C_Y \), \( C_{\text{roll}} \), \( C_n \).
  Each of these cells contains the mean and standard deviation values for this coefficient, for this site, in this data set.

Input parameters

- data: it is the compulsory input parameter. It can be:
  - the original data, a cell array containing one data set per cell.
  - the fused data, a CEASIOM formatted structure, containing, just like the original data sets, important information about the plane, as well as two aerodynamic tables, these tables being, in this case, the result of a fusion. Just like any computer generated data, outliers might have appeared during the fusion process and filtering them out will increase its overall accuracy, preventing incoherent results to appear after the SDSA module calculations.

The body of the function is divided into 3 main parts:

→ The first part consist in two operations:
  - dividing up of the data by set and sites,
Calculating and storing the mean and standard deviation for each aerodynamic coefficient, for each site and for each data set.

This is done according to the nature of the input parameter:

- if `data` contains the original data sets, it is a cell array indicating, among other information, the size of each site in each data sets.

- If `data` is a data set resulting from the fusion, it is a structure array. It contains, just like the original data sets, who are also CEASIOM-formatted, the size of each of its sites, but since it is a structure array, the dividing up will be coded differently than for a cell array.

- if the input data is neither a cell nor a structure array, an error message is displayed: "This function is defined for cell and double arrays inputs only".

Using these information, the data is divided up by data sets and site. Once this is done, the mean and standard deviations, noted `std`, are calculated for each site and data set, and stored into the `stats` cell array.

→ The second part consist in the spotting and storing of the suspicious and outliers points. For each site and for each data sets, all points that match the condition $3 \times std > x > 2 \times std$ are stored into the `suspicious` cell array, while those matching the condition $x > 3 \times std$ are stored in the `outliers` cell array.

This part also takes care of the plotting of all the categories of points, with a simple but talking colour code and easy markers.

→ The last part consists in formatting the `outliers` and `suspicious` cell arrays in order to make them usable as inputs to the filtering function.

At the end of part 2, these arrays are columns the size of the site considered, that contain 1 where an outlier/suspicious point is located, and 0 elsewhere. In order to be usable by the filter function, the third part replaces each of the 1 by the number of their line, and deletes the 0.

III.1.2.3 Output

The output for this function is the same for all input parameters. It produces 8 figures, for each of the 8 sites.

Each figure is divided in n lines, n for the number of data sets, and 6 columns, for each aerodynamic points. Below is one figure of a typical output for the `myvisual` function for a data input of two data sets:
Figure 17: Illustration of one of the output figures for myvisual.

As can be seen on the figure, each subplot on a line represents an aerodynamic coefficient for a data set. Each one of them contains 7 horizontal lines.

- The red one represents the mean for this aerodynamic coefficient, for this site, in this data set.
- The green lines represent $\text{mean} + \text{std}$, $\text{mean} + 2\times\text{std}$, $\text{mean} + 3\times\text{std}$ and $\text{mean} - \text{std}$, $\text{mean} - 2\times\text{std}$ and $\text{mean} - 3\times\text{std}$.

The points are divided up in three colors for each category:

- the blue points: the valid points, matching the $\text{mean} - 2\times\text{std} < x < \text{mean} + 2\times\text{std}$ condition.
- the red points: the suspicious points, matching the $\text{mean} + 3\times\text{std} > x \geq \text{mean} + 2\times\text{std}$ or $\text{mean} - 2\times\text{std} > x \geq \text{mean} - 3\times\text{std}$ conditions.
- the black points: the outliers, matching the $x \geq \text{mean} + 3\times\text{std}$ or $x \leq \text{mean} - 3\times\text{std}$ condition.

Each subplot is titled with the number of the dataset it belongs to. Each subplot has a legend with the name of the aerodynamic coefficient whose values are displayed.

Using this function, the engineer can clearly visualize the points and judge whether they should be filtered out or not.

In this very example, the spotted points are all very distant from the mass of the valid points. They should be filtered out.

In the following example, the suspicious points seem, in many cases, quite coherent with the rest of the data. It is up to the engineer to decide whether or not they should be filtered.
Figure 18: Illustration of outliers and suspicious points possibly coherent with the pattern of the data.
III.2 Treatment

Motivation
After the data has been verified using the myplot and the myvisual functions, the second step is to treat the data.
The apparition of outliers in computer generated data is a common problem. These points have to be filtered in a way that suits the engineer. Once the outliers are treated, a third step remains.
As explained earlier, certain jumps may appear for some of the aerodynamic coefficients. It is absolutely normal. However, due to the architecture of the kriging functions, they may induce errors in the calculations during the fusion.
For better understanding, an example of the problem is given below.

![Illustration of the purpose of the mapping process](image)

**Figure 19:** Illustration of the purpose of the mapping process

The green points represent values from a data set, for a given aerodynamic coefficient. A jump in the values can be seen.
As part of the fusing process, the values in the jump need to be interpolated. As explained earlier, this is done by kriging, where, in order to interpolate a value \( f(x) \) for a given \( x \), the function will consider the known \( [x_1, x_2 \ldots x_n] \) that are considered close to it.
This notion of "close" is given by what is called the interpolation length. It is the radius of a circle around \( x \). The \( x_i \) enclosed in that circle are considered close. This is an intuitive value. It is not given by a formula. However, it can not be too small, in order to maintain a certain degree of correlation between the values of a curve, nor too big, for then the whole curve is rendered useless: it becomes a mean of all the values in the curve. In these circumstances, the interpolation length always englobes a certain number of \( x_i \). In the schematic, \( x \) is shown by the black circle. The interpolation length is materialized by the magenta semi-circle.
As can be seen, the values taken into consideration are in majority located in the two plateau respectively preceding and following the jump. This will cause small errors, since for good accuracy, the values around the jump should be taken more into account than the ones far away from it. To avoid this problem, a custom-mapping had to be developed as part of the pre-treatment of the data.
III.2.1 myfiltermap

III.2.1.1 Purpose
The goal of this function is to carry out the complete pre-treatment of the data sets, by filtering and mapping them.
As presented in the previous section, a common problem with computer generated data is the appearance of outliers. The function myvisual allows the engineer to visualize these points, as well as those who are possibly inaccurate, called ”suspicious” points.
After visualizing them in relation with the other valid points and the overall values for a given aerodynamic coefficient, the engineer can decide whether to filter out the outliers, the outliers and the suspicious points, or to not take any action.
This is done with the myfiltermap function.

In addition to the filtering, the myfiltermap function also realizes a custom-made mapping, where jumps in the function appear.
As explained earlier, the jumps that appear in the data can lead to inaccuracies due to the way kriging works. No matter how small the correlation length is chosen, there will always be points used in the interpolation that belong to the two plateaux located before and after the jump. These will affect the accuracy of the interpolated values in the jump.
In order to reduce the number of values from the plateaux used in the interpolation, a solution is to create a custom-made mapping, that will ”stretch” the jump, so that the plateaux are moved away from the location of the jump.
This mapping function must thus carry out two tasks:
→ detect eventual jumps
→ realize the stretch

This is illustrated in figure 21 below. In this figure, $x$ represents any of the control or state parameters, while $C_x$ is any of the 6 aerodynamic coefficients.

![Figure 20: Illustration of the effect of the mapping.](image)

The way the mapping should work is illustrated in this figure: a jump is detected, at $x = 5$, and then stretched out significantly. It can be seen that only the $x$ values are stretched, while the $C_x$ values remain unchanged. This is important to maintain the accuracy of the process: by stretching the $x$ values, the kriging process will work optimally, taking into account less values from the plateaux, but using the right $C_x$ values for the interpolation.

The function was written to detect all jumps, positive or negative, and to act on them accordingly.
III.2.1.2 Code description

The function is declared as:

\[ \text{[treated\_data, map]} = \text{myfiltermap(data, outliers, suspicious, opt1, opt2)} \]

where:

Input parameters

- **data**: this is the cell array containing the original data sets.
- **outliers**: this is the cell array containing the position of the points spotted as outliers. It is one of the outputs of the `myvisual` function.
- **suspicious**: this is the cell array containing the position of the points spotted as suspicious. It is one of the outputs of the `myvisual` function.
- **opt1**: this parameter indicates which type of filtering the engineer wants to have completed. It can take 3 values:
  - "0": No filtering will be carried out. The outliers and suspicious points are kept.
  - "1": The outliers and suspicious points will be filtered out.
  - "2": Only the outliers will be filtered out. The suspicious points are kept.
- **opt2**: this parameter indicates whether the engineer wants to have the data mapped or not. It can take 2 values:
  - "0": No mapping will be carried out.
  - "1": The data will be mapped.

Output parameter

- **treated\_data** is a structure containing the filtered and mapped data, formatted in a suitable way for the fusion. It is divided in 8 cells, for each site, 1 structure, containing the final size of each site after filtering and 1 double indicating the number of data sets originally involved.
- **map** is a cell array containing the mapped parameters, arranged by data set, site, and parameters.

The body of the function is divided into three main parts:

→ The first part is, as in the previous functions, the dividing up of the data by data set and sites. Since the input parameter can only be a cell array containing the original data sets, this is completed by simply using the \( N_X \).

→ The second part is the filtering.
   A **switch** on the value of **opt1** determines what actions will be taken:
   - if **opt1**=0, the program simply displays the message: "No filtering".
   - if **opt1**=1, the outliers and suspicious points need to be filtered out. This is done in three separate steps, one in this part of the code, and the other two in the third part of the code regarding the mapping.
     The **outliers** and **suspicious** cell arrays indicate precisely the location of the points to filter out: they are separated by data set, site and aerodynamic coefficient.
     With these pieces of information, the outliers and suspicious points are all replaced by **Not-a-Number**—NaN—values. This notation represents an undefined or unrepresentable value.
– if opt1=2, the outliers must be filtered out.
This is done in the same manner used in the case opt1=1: using the outliers cell arrays, the undesirable points are located and replaced by NaN values.

\[ \text{→ The third part relates to the mapping.} \]
A switch on opt2 decides of the course of action:

– If opt2=0, the data won’t be mapped. The program displays the message: "No mapping".
However, the output of the myfiltermap function is a structure formatted for the fusion process.
Even though the data has not been mapped, and possibly not filtered either, this formatting is carried out directly into the mapping part of the function, differently for both values of opt2.

– if opt2=1, the data will be mapped.
This implies several operations.
The first step is to arrange the data to be tested for jumps.
As shown in the previous figure, in order to see the jumps properly, the aerodynamic values must be a function of only one parameter. In the case of a three parameter site, for instance \([\alpha Mach \beta]\), the values are presented as shown below:

\[
\begin{array}{cccccccc}
\alpha & M & \beta & C_L & C_D & C_m & C_Y & C_{roll} & C_r \\
\hline
\alpha_1 & Mach_1 & \beta_1 & x & x & x & x & x & x \\
\alpha_2 & Mach_1 & \beta_1 & x & x & x & x & x & x \\
\alpha_3 & Mach_1 & \beta_1 & x & x & x & x & x & x \\
\alpha_1 & Mach_2 & \beta_1 & x & x & x & x & x & x \\
\alpha_2 & Mach_2 & \beta_1 & x & x & x & x & x & x \\
\alpha_3 & Mach_2 & \beta_1 & x & x & x & x & x & x \\
\alpha_1 & Mach_1 & \beta_2 & x & x & x & x & x & x \\
\alpha_2 & Mach_1 & \beta_2 & x & x & x & x & x & x \\
\alpha_3 & Mach_1 & \beta_2 & x & x & x & x & x & x \\
\alpha_1 & Mach_2 & \beta_2 & x & x & x & x & x & x \\
\alpha_2 & Mach_2 & \beta_2 & x & x & x & x & x & x \\
\alpha_3 & Mach_2 & \beta_2 & x & x & x & x & x & x \\
\end{array}
\]

\[ \text{Figure 21: Illustration of the architecture of a three-parameter site aerodynamic table.} \]

In order to test the jumps, the tested values of aerodynamic coefficients must be function of a single parameter, meaning one parameter varies, while the two others remain constant.
To facilitate this testing, the data is rearranged and stored in new tables. In the example above, x different tables are obtained:

* For \(C_x\) function of \(\alpha\):

\[
\begin{array}{cccccccc}
\alpha & M & \beta & C_L & C_D & C_m & C_Y & C_{roll} & C_r \\
\hline
\alpha_1 & Mach_1 & \beta_1 & x & x & x & x & x & x \\
\alpha_2 & Mach_1 & \beta_1 & x & x & x & x & x & x \\
\alpha_3 & Mach_1 & \beta_1 & x & x & x & x & x & x \\
\alpha_1 & Mach_2 & \beta_2 & x & x & x & x & x & x \\
\alpha_2 & Mach_2 & \beta_2 & x & x & x & x & x & x \\
\alpha_3 & Mach_2 & \beta_2 & x & x & x & x & x & x \\
\end{array}
\]

\[ \text{Figure 22: Tables obtained during the mapping for an [\alpha Mach \beta] site - } C_x=f(\alpha) \]

* For \(C_x\) function of the Mach number:

\[
\begin{array}{cccccccc}
\alpha & M & \beta & C_L & C_D & C_m & C_Y & C_{roll} & C_r \\
\hline
\alpha_1 & Mach_1 & \beta_1 & x & x & x & x & x & x \\
\alpha_2 & Mach_2 & \beta_1 & x & x & x & x & x & x \\
\alpha_1 & Mach_1 & \beta_1 & x & x & x & x & x & x \\
\alpha_2 & Mach_2 & \beta_2 & x & x & x & x & x & x \\
\alpha_3 & Mach_2 & \beta_2 & x & x & x & x & x & x \\
\end{array}
\]
The process of rearranging the data to obtain all these tables is done automatically on all sites for all data sets. The tables obtained are stored in a new cell array, separated by data sets, then by sites, and finally by varying parameter. Once this is complete, the mapping itself can begin. Two operations are carried out for each of the tables:

* Detecting the jumps.
  This is done by test on the slopes, one aerodynamic coefficient at a time. In order to detect both positive and negative jumps, the test is carried out on the absolute values of the slopes. If the table has three points or more, the slope between point \( x + 2 \) and \( x + 1 \) is compared to the slope between the points \( x \) and \( x + 1 \). If one of the absolute values is higher than \( \epsilon \) by default—times the absolute value of the other one, a jump is considered to happen in \( x + 2 \). The height of the jump is also measured, taking the difference between \( C_x = x + 1 \) and \( C_x = x + 2 \). These values are stored in another table. If the table has less than three points, it is impossible to detect a jump. No jump is returned.

* Stretching the values.
  Using the locations of the jumps, \( x = x + 2 \), and the height of the jumps, a mathematical formula is applied to the data, realizing the stretch. The formula is the following:

\[
P = x + \Delta \ast (\tanh\left(\frac{x - x^*}{\delta}\right) + 1/4); \quad (13)
\]

where:

- \( P \) is the mapped parameter. It will be the only quantity "seen" by the fusion function as a site parameter.
- \( x \) is the site parameter, for instance, \( \alpha \), the Mach number, \( \beta \), or any of the state or control parameters.
- \( \Delta \) is the height of the jump.
- \( x^* \) is the location of the jump.
- \( \delta \) is the width of the jump.

The mapped parameters are then stored in a new cell array, \texttt{map}, alongside the parameters that remained constant for the calculation. They are stored with labels indicating which data set, which site, and which parameter they relate too.

Once this is done for all parameters in all sites, the data must be reconstructed and formatted for the fusion.

This means doing the reverse process than the step before: from the multiple tables, now containing the treated data, reconstruct an ordered table for each site.

To format the data properly for the fusion, the first step is to remove all the NaN values. They have a disastrous effect on the fusion process: the program, unable to identify or use these values, will not ignore them, but displays an error message, preventing the completion of the fusion.

This could have been done quite easily at the filtering stage of the process; however, by keeping these values, the process of mapping is made much more easier and reliable. It would be different if the outliers and suspicious points were simply removed at the filtering stage. Indeed, this would mean removing seemingly random values inside the tables. During trial stage, the gaps in the values proved very hard to work around when rearranging the data to obtain mapping tables. These tables are automatically generated, and though it was be possible to recuperate all the values for the different tables, it proved a long and unnecessary trouble, when the NaN values are ignored during mapping, thanks to the way it is coded. By letting these values in the data after filtering, precious time is gained in the reconstruction.

However, these values need to be erased before the fusion. The first step when formatting the data is thus to delete them.

In order to remain coherent in the size of the samples, if a value of \( C_L \) is a NaN, all the values coming as a result of this site are deleted as well. This is also a potential security since a calculation that has gone wrong on one side may have gone wrong on another, less apparent way. Once all lines containing NaN are deleted, the actual formatting starts.

### III.2.1.3 Output

The output structure has a very specific format, so that it can be used directly for the fusion. As explained in the III.2.1 section, the main function used to perform the fusion is the \texttt{dacefit} function.

The two main input parameters are two matrices, named \texttt{S} and \texttt{Y}.

The \texttt{S} matrix contains the sites, while the \texttt{Y} matrix contains the values of the aerodynamic coefficients associated to the sites in the \texttt{S} matrix. The \texttt{S} matrix must contain all the sites, in all data sets, positioned end to end. The same is done for the matrix \texttt{Y}: it must contain all the values of the aerodynamic coefficients of the different data sets, end to end.

As explained earlier in this section, \texttt{treated_data} contains 8 cells for each site. Each of these cell arrays contains 6 cells, one for each aerodynamic coefficient. They follow the same format: the three first columns contain the filtered and mapped data for the three site parameters. The next three columns contain the "reference" values for the three sites. The reference values are the original values of the sites parameters, before mapping.

This formatting is carried out even when the engineer has chosen not to filter or map the data so that the output is properly formatted to be directly used by the fusion function.

#### Analysis

Below is an example of mapping from the function:
As can be seen on the figure, it presents inaccuracy problems: the stretch, though more important in the jump, is actually realized on all parts of the curve, which could lead to a lack of precision that it was coded to avoid on another level. In some cases, it has even been noticed that the function merely translates the curves, without even stretching the jump.

By translating the values for one data set in this manner, the values used in the kriging interpolation might take into account values of the other data sets that due to the translation end up being close while they are actually not. This would increase even more the inaccuracy of the obtained data.

In this regard, a possible solution would be to map the data after assembling it into a single data set, so that all values are mapped, to maintain a distance coherent with the mapping between all the values.

The overall inaccuracy problem could certainly be solved by refining the formula used to realize the stretch, so that only the values on the stretch are affected.

### III.2.2 Post-fusion treatment: myfilterfused

#### III.2.2.1 Purpose

The fusion is a computer data generating process, and as for all these types of processes, outliers can appear among the data. As such, it might be interesting to filter them out to obtain an even more accurate data set.

Using first the myvisual function with the Table array of double—output of the myfusion function containing the aerodynamic data resulting from the fusion—as input to obtain the outliers and suspicious cell arrays associated, the fused data can be filtered using the myfilterfused function. Since the data has been fused already, no mapping is needed.

The myfiltermap function could not be used for two reasons:

- the specific formatting of its output, in order to make the data ready for the fusion. In this case, the data is already fused and its format must not be changed.

- the problem of the filtering, that is, as explained in section III.4.1, only partial in the filter part of the myfiltermap function, since it replaces the targeted values by NaN; only in the mapping part are the values actually deleted.

In order to use the same function to perform these tasks, options completely new and specific to this part of the process would have to be coded, while making for a messy code. In the light of these considerations, I opted for a new function.
III.2.2.2  Code description
The function is declared as:

\[
[\text{Table}, \text{Fused\_data\_treated}] = \text{myfilterfused}(\text{data}, \text{outliers}, \text{suspicious}, \text{opt})
\]

where:

- **data** is the output from the `myfusion` function. It is a CEASIOM formatted structure containing the aerodynamic tables coming from the fusion.
- **outliers** is one of the outputs of the `myfiltermap` function. It is a cell array containing the location of the points spotted as outliers, by site and aerodynamic coefficient.
- **suspicious** is one of the outputs of the `myfiltermap` function. It is a cell array containing the location of the points spotted as suspicious, by site and aerodynamic coefficient.
- **opt** indicates which filtering the engineer wants to have completed. `opt` can take two values:
  - "1": the outliers and suspicious points will be filtered out.
  - "2": only the outliers will be filtered out.

Output parameters

- **Fused\_treated\_data** is a CEASIOM formatted structure containing, among all the other information relative to the plane, the fused and filtered aerodynamic data.

The body of the function is divided into three main parts.

- The two first parts are the same in essence as the ones in the `myfiltermap` function: the first part divides up the data by sites, while the second handles the filtering, immediately deleting the values to filter.

- The last part reassembles the data into `Table` and `Fused\_data\_treated`. The argument could be made that the values of the aerodynamic tables in `Fused\_treated\_data` are the same as in `Table`, and that there is thus no point in adding the `Table` output. However, `Table` is used by the `myplot` function. In the form of an already arranged unique table containing all the data, it is more easily manipulated than the CEASIOM formatted structure, containing two separated aerodynamic tables.

III.2.2.3  Output
The main output of the function is `Fused\_treated\_data`, a CEASIOM formatted structure containing the fused and filtered data. It is saved in the current Matlab Workspace, but would the engineer need to stop the process and come back to it later, the structure is also saved as an .xml file to be loaded at anytime by the SDSA module.
With `Table`, the engineer can also visualize the data, and compare it to the treated data or to the raw data used in the fusion process.
III.3 Data fusion

III.3.1 Purpose

This function carries out the actual fusion process. Its purpose has been heavily discussed in the previous sections, so there is no need to repeat it here.

Once the data has been treated, the myfusion function will proceed to the fusion process, creating new, more detailed and complete, aerodynamic tables.

This function also creates a new .xml file, compatible with the SDSA CEASIOM module.

III.3.2 Code description

The function is declared as:

\[
[\text{Table, Fused\_data}] = \text{myfusion}(\text{treated\_data})
\]

where:

- **Input parameters**
  - **treated\_data** is a structure from the myfiltermap function, containing the potentially filtered and mapped data from the original data sets. It has been specially formatted to be used by this function.

- **Output parameters**
  - **Fused\_data** is a structure of CEASIOM format, containing the aerodynamic tables from the fusion, as well as the essential information for the SDSA analysis.
  - **Table** is an array of double containing both fused aerodynamic tables from the fusion process. It has the following format:

```
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline
\hline
\text{\(\alpha\)} & \text{\(M\)} & \text{\(\beta\)} & \text{\(q\)} & \text{\(p\)} & \text{\(r\)} & \text{\(\delta_x\)} & \text{\(\delta_y\)} & \text{\(CL\)} & \text{\(CD\)} & \text{\(Cm\)} & \text{\(CY\)} & \text{\(C_{roll}\)} & \text{\(C_{L}\)} \\
\hline
x & x & - & - & - & - & - & - & x & x & x & x & x & x \\
\hline
x & x & x & - & - & - & - & - & x & x & x & x & x & x \\
\hline
x & x & - & x & - & - & - & - & x & x & x & x & x & x \\
\hline
x & x & - & - & x & - & - & - & x & x & x & x & x & x \\
\hline
x & x & - & - & - & x & - & - & x & x & x & x & x & x \\
\hline
x & x & - & - & - & - & x & - & x & x & x & x & x & x \\
\hline
x & x & - & - & - & - & - & x & x & x & x & x & x & x \\
\hline
\end{tabular}
```

Figure 26: Illustration of the format of a fused aerodynamic Table

While the content of **Table** appears in the CEASIOM traditional format of two distinct tables in Fused\_data, Table is an easier quantity to manipulate by the myplot and filterfuse functions.

The body of the function is divided into three main parts:

- The first part sets up a few new parameters, needed to obtain a physically meaning table at the end of the fusion process.
  
  In the fusion process, after all multiple sites have been merged, the new \(S\) and \(Y\) matrices, containing respectively the sites and the corresponding values of the aerodynamic coefficients, are given to the dacefit function. With these inputs, it builds a model of the function \(f\) such that \(Y = f(S)\). This model is then given to the predictor function, along with another input parameter: \(X\). \(X\) is a vector containing the values for which the output will be predicted. It must take successively all the values contained in the data sets, for all combinations of parameters.

  The first step at the beginning of the myfusion function is to obtain all the values taken by
each parameters in all the data sets put together. This is done by using the Matlab function `unique`, and the values are stored in cell arrays to be used by the actual fusion process in part two of the function. These cells will however also include the values that result from the mapping.

As part of the mapping process, some values might have been modified slightly, if located at proximity of a jump. However useful to the interpolation process at the heart of the fusion, these values do not have any physical meaning. In the final table containing the fused data, it is important that the actual, unmapped, values appear.

To realize this, each mapped value of a site parameter must be associated to its original one. Using the unchanged values stored in `treated_data`, and the result of the previous step, each value is associated to its original. These values are stored in new cell arrays, with names `x_model_z`, where `x` is the name of the parameter whose original values are being fetched, and `z` the name of the site the parameter belongs to.

→ The second part is the fusion itself.

It is carried out site by site, and the values are stored end to end in the format presented above. Each site fusion follows the same pattern:

A message indicating which site is being fused is displayed: "Fusing dat sets for [α Mach z]...", `z` being the third parameter, or nothing for the first site [α Mach].

Inside a loop on the number of aerodynamic coefficients, 6, the S and Y matrices are given to the `dsmerge` function. In this case, the S and Y matrices are respectively:

- `treated_data.z(j)(:,1:2)` and `treated_data.z(j)(:,5)` for the [α Mach] site,
- `treated_data.z(j)(:,1:3)` and `treated_data.z(j)(:,7)` for the other site sites.

where:

- `j` is the loop parameter, going from 1 to 6.
- `z` indicates the site: Alpha, Beta, q, p, r, elev or ail.
- the columns 1 to 2 contain the α and Mach number values, column 3 contains the third parameter.
- column 5, for the [α Mach] site, contains the values of the targeted aerodynamic coefficient. Column 7 serves the same function for three parameter sites.

The output of the `dsmerge` function is given to the `dacefit` function, that, from it, builds an interpolation model for the data.

Inside multiple loops on each parameter of the fused site, the model is given to the `predictor` function, along with the X vector, containing the values of each parameter of the site.

As an example, if there are a total of 3 values for the α parameter, 2 for the Mach number parameter and 2 for the β parameter, X will successively take, for the [α Mach β] site, the following values:

\[
\begin{bmatrix}
[\alpha_1 \text{ Mach}_1 \beta_1], [\alpha_2 \text{ Mach}_1 \beta_1], [\alpha_3 \text{ Mach}_1 \beta_1], [\alpha_1 \text{ Mach}_2 \beta_1], [\alpha_2 \text{ Mach}_2 \beta_1], [\alpha_3 \text{ Mach}_2 \beta_1] \\
[\alpha_1 \text{ Mach}_1 \beta_2], [\alpha_2 \text{ Mach}_1 \beta_2], [\alpha_3 \text{ Mach}_1 \beta_2], [\alpha_1 \text{ Mach}_2 \beta_2], [\alpha_2 \text{ Mach}_2 \beta_2], [\alpha_3 \text{ Mach}_2 \beta_2]
\end{bmatrix}
\]

The predictor function gives the interpolated value for each of the input sites X, and the table is gradually filled.

In this process, each value for the site parameters is stored in the table by its original value. This causes the apparition of multiple sites.

As an example, let’s assume that the Mach number originally has 3 values: \(M_1 = 0.2\), \(M_2 = 0.3\) and \(M_3 = 0.6\).

Due to the mapping, some values of \(M_1\) have been mapped, and there are \(M_4 = 0.31\) values.

The fusion is going to be carried out by order of increasing Mach numbers. It will start with \(M_1 = 0.2\), then \(M_2 = 0.3\), \(M_4 = 0.31\), and so on, but in the table, all values will be
tagged as their original, physical value, which means the values for \( M_4 \) will be saved as values for \( M_1 \), as presented above:

\[
\begin{array}{c|cccc|cccccccc}
\alpha & M_1 & \beta & q & p & r & \delta_e & \delta_y & \delta_a & C_L & C_D & C_m & C_Y & C_{roll} & C_a \\
\hline
\alpha_1 & - - - - & - - - & x & x & x & x & x & x & x & x & x & x & x & x \\
\alpha_2 & - - - - & - - - & x & x & x & x & x & x & x & x & x & x & x & x \\
\alpha_1 & M_2 & - - - - & - - - & x & x & x & x & x & x & x & x & x & x & x \\
\alpha_2 & M_2 & - - - - & - - - & x & x & x & x & x & x & x & x & x & x & x \\
\alpha_1 & M_1(M_4) & - - - - & - - - & x & x & x & x & x & x & x & x & x & x & x \\
\alpha_2 & M_1(M_4) & - - - - & - - - & x & x & x & x & x & x & x & x & x & x & x \\
\hline
\end{array}
\]

**Figure 27:** Illustration of the ”multiple site” phenomenon in a fused table.

However, there can not be any multiple site for the coherence of the data to pass on to SDSA. This is the reason why the last step is to automatically go through the in-progress table and spot these multiple sites. They are then merged in a single one, by doing the mean of all their values for each aerodynamic coefficient. The size of the new in-progress table is then calculated and given to the next site to be fused, so that the values follow each other properly from one site to the other.

The third part consists of calculating and assembling the data needed to realize a CEASIOM formatted output, here, the **Fused_data** structure.

A CEASIOM formatted structure contains:

- Information on the plane’s geometry, pilot and engines,
- the number of values taken by each parameter of the state and control vectors,
- the aerodynamic tables.

The information on the geometry and engines of the plane, as well as on the pilot, are the same for all data sets involved in the fusion, since the airplane is the same. These information can be copied from one of the data sets.

However, since the fused data set encompasses all the data contained in the data sets used in the fusion, the number of variation for each parameter is the sum of the \( N_x \) numbers of each data set for each parameter. Using the `unique` and `numel` Matlab function, these value can simply be retrieved from each column of the table containing the fused data. This is done, and saved under the proper \( N_x \) names.

The global table containing the fused data must also be cut in half, to obtain the CEASIOM formatted tables containing the state parameters and the control parameters— save \( \alpha \) and Mach — separately.

Using again the `unique` function to localize the line where the first values of \( \delta_e \) appear, the table can be cut in two and rearranged properly.

Finally, the newly built structure is saved as a .xml file to be used by SDSA.

### III.3.3 Output

The function produces two outputs:

- the CEASIOM formatted **Fused_data** structure, saved in the MATLAB Workspace, as well as an .xml file. The engineer can use the one saved in the Workspace to work with SDSA, or if the process needs to be interrupted, load later on the .xml file to get started with SDSA directly.

- the **Table** double array containing the aerodynamic table from the fusion. It is easier to manipulate by the `myplot` function, to plot the treated data and the fused data, by the `myvisual` function when spotting the outliers and suspicious points for the newly fused datan, or by the `myfilterfuse` function.
Though the function was not tested as yet on actual data, the test results on artificial data seem good and coherent.

In this function, all data sets are considered equally for the interpolation of the values. However, since these sets present different degrees of fidelity, a way to further improve the accuracy of the process would be to assign different weight to the data sets, function of the degree of fidelity with which they were generated.

By doing so, the highest fidelity data sets would impose the trend for the values, thus making the whole more accurate.

Figure 28: $C_{roll}$ for [$\alpha$ Mach $\beta$] site, treated and fused data compared
III.4 The Graphic User Interface

III.4.1 Purpose

There are two main reasons to the implementation of the complete fusion process into a GUI:

→ The first one is to help the engineer in his task. The whole Fusion process is composed of 6 different functions:

- a function loading the .xml files into a single cell array containing one data sets in each cell.
- myplot:
  myplot(data,opt,varargin)
- myvisual:
  [outliers,suspicious,stats]= myvisual_gui(data)
- myfiltermap:
  [treated_data,map]= myfiltermap(data,outliers, suspicious, opt1, opt2)
- myfusion:
  [Table,Fused_data] = myfusion(treated_data)
- myfilterfused:
  [Table,Fused_data_treated]=myfilterfused(data, outliers, suspicious, opt)

These functions have to be used in a very specific order, and several of the outputs are inputs to the next function. Their implementations into an intuitive GUI makes it all the more simpler to fuse data properly.

→ The second main reason is that this process is part of the CEASIOM software, and must communicate with the other CEASIOM modules, especially SDSA. In this optic, the functions had to be implemented into a GUI matching, and integrated into, the CEASIOM interface.

III.4.2 Description

Layout

The Fusion GUI was implemented using GUIDE: The Matlab GUI Development Environment, to remain coherent with the CEASIOM GUI, also implemented in Matlab.

The main window for the Fusion GUI is presented in figure 27. It is divided into three main areas:

→ The top part of the window contains the commands for the basic functions. It consists of three push buttons:

- The ”LOAD DATA” button: upon click, a dialog box opens—cf figure 28—, displaying the list of files present in the current directory and prompting the user to ”Select a file”. There, the engineer can select as many .xml files as needed for the fusion he wants to carry out. Clicking ”OK” will load the chosen data sets into a single cell array, each cell containing one of the chosen data set, the first chosen in cell 1, the second in cell 2 and so on. If the engineer has accidentally selected a non .xml file, an error will appear but the program will not crash. The engineer can then click on LOAD DATA again and select the correct files.

A confirmation message is then displayed after the ”OK” button is pressed: ”Loading Completed”. Once closed, a new dialog box opens, asking the user whether the data should be stored in the current Matlab Workspace, and with what name. The name by default is ”data”. If chosen, the data is then exported to the Matlab Workspace, and the dialog box disappears.

The loading of the data sets activates parts of the Visualization panel.
– The "RESET" button, as its name indicates, sets the program back to the start. Internally, it empties all structures of their values, and disable all buttons that are not enabled in the start configuration.

– The "Close Plots" button becomes active once something is plotted. Clicking it will close all open plots, and becomes inactive again, till new figures are plotted.
The middle part is the "Visualization" part. It contains the panels tied to the `myplot` and `myvisual` functions.

- The upper part of the panel contains the commands to the `myplot` function. It is made of:
  * 10 check boxes: 8 for each site, 1 for \( C_D = f(C_L) \), and 1 for all of the previous choices. It indicates to the function which site the user would like to plot. It is translated in the code in terms of the `opt` input parameter. `opt = 1` if the "Basic" check box is ticked, 2 if the Beta check box is ticked and so on, according to the code of `myplot_map`. It is possible to check several boxes, and all the targeted sites will be plotted.
  * 3 radio buttons: "Original data", "Treated data" and "Fused data". They serve to indicate which data is going to be plotted.
  * 1 push button: "Plot". On click of this button, the selected sites will be plotted in a single click.

Upon completion of the plotting, a message box is displayed indicating: "Plotting Complete".

- The lower part of the panel contains the commands to the `myvisual` function. It is made of:
  * 2 radio buttons: "Original data" and "Fused data", indicating which data the outliers and suspicious are being visualized for.
  * 1 push button: "View outliers & suspicious points". On click of this button, the figures presenting the outliers and suspicious points will be plotted.

Upon completion of the plotting, the message box appearing at the end of the plotting is displayed, indicating that the plotting is complete.

A dialog box then appears, asking the engineer whether the three outputs of the function—the `outliers`, `suspicious` and `stats` cell arrays—should be exported to the current Matlab Workspace. Defaults names are proposed, but they can of course be edited, directly in the dialog box. The engineer can choose to export one of two out of the three by unticking/ticking the corresponding boxes.

![Save to Workspace dialog box](Image)

**Figure 31:** Message box for the exporting of several arrays to the Matlab Workspace.

The engineer can choose to do so with all, or some of them, or to not do it. It does not affect the rest of the process, as the cell arrays are stored internally in the program.

The bottom part is the "Actions" part. It contains the commands to the `myfiltermap` function, `myfusion` and `myfilterfused` functions. The panel is divided into the "Pre-Treatment" subpanel, and the commands related to the fusion and the fused data.

- The "Pre-Treatment" panel contains:
* 2 pop-up menus:
  - One listing the options for the filtering: "Filtering1", "Filtering2", "No Filtering", corresponding respectively to $\text{opt1} = 1, 2 \text{ or } 0$,
  - One listing the options for the mapping: "Mapping" or "No mapping", corresponding respectively to $\text{opt2} = 0 \text{ or } 1$.
* 1 push button: "OK".
  Upon clicking it, the filtering and mapping process starts, and a message box appears, stating "Treatment in progress".

When the treatment is finished, a message box appears indicating that the process has been completed, and a dialog box asks whether the outputs — the map cell array and the treated data structure — should be exported to the Matlab Workspace.

- The fusion part contains:
  * 1 push button "Fuse data", which upon clicking starts the fusion process. A message box is displayed indicating: "Fusion in progress". When the fusion is finished, another message box appears indicating: "Complete - Fused data saved as Fused_data.xml".
  * 1 pop-up menu listing the options for the filtering of the fused data: "Filter fused 1" or "Filter fused 2", corresponding respectively to the input parameter $\text{opt} = 1 \text{ or } 2$ for the $\text{myfilterfused}$ function.
  * 1 push button "OK", that starts the filtering of the fused data. A message box appears indicating: "Treatment in progress". Once the treatment is finished, another message box appears: "Treatment completed - File saved as Fused_treated_data.xml".

There are an additional two push buttons in the top right corner:

- An "About" button, giving basic information on the program:
  ![About message box](image)

  **Figure 32:** About message box

- A "Help" button, giving a few indications on the filtering options and the interpretation of the color code in the $\text{myvisual}$ plots:
  ![Help message box](image)

  **Figure 33:** Help message box.

—

There are an additional two push buttons in the top right corner:

- An "About" button, giving basic information on the program:

  ![About message box](image)

  **Figure 32:** About message box

- A "Help" button, giving a few indications on the filtering options and the interpretation of the color code in the $\text{myvisual}$ plots:

  ![Help message box](image)

  **Figure 33:** Help message box.
Features
Once the layout was finished, and the functions associated to the relevant buttons, a few mechanisms had to be coded in order to ensure coherence to the GUI as a whole, and to help the user avoiding making mistakes in the process.
In this optic:

→ All buttons but Load data, RESET, Help, and About, are inactive when the GUI is launched.
→ Once data is loaded:
  – the plot section becomes active, with the radio button “Original data” selected by default.
  – the ”Treated data” and ”Fused data” radio buttons are inactive.
→ If the Plot button is clicked on but no site is selected, a warning message box appears indicating:

![Warning message](image)

**Figure 34:** Warning message appearing when no site to plot is selected and ”Plot” is clicked on.

→ If the Plot button is clicked but no data to plot is selected— the user can have accidentally unselected the ”Original data” button—, a warning message box appears indicating:

![Warning message](image)

**Figure 35:** Warning message appearing when no data to plot is selected and ”Plot” is clicked on.

→ Once the plotting is complete:
  – the visualization of the outliers and suspicious points section is enabled, with the ”Original data” radio button selected by default.
  – the ”Fused data” radio button is inactive.
  – the ”Close plots” button becomes active. When clicked on, the open plots are closed, and the ”Close plots” button becomes inactive again.
→ Once the plotting of the outliers and suspicious points is completed:
  – the Pre-Treatment subpanel is enabled,
  – the "Close plots" button is enabled. When clicked on, all open plots are closed and the "Close plots" button is disabled again.

→ Once the pre-treatment is done:
  – the "Fuse data" push button is enabled,
  – the radio button "Treated data" in the plot section is enabled.

The user can then plot the Original data as well as the filtered and mapped one.

→ When the fusion is complete:
  – the "Fused data" radio button in the outliers and suspicious section is enabled, so that the user can visualize them for the fused data.
    If the "Fused data" radio button is selected, the "Original data" is deselected and vice-versa
  – The "Fused data" radio button in the plot section is also enabled.
    It can only be selected if the "Treated data" radio button is selected too.
    It is possible to select all three radio buttons: "Original data", "Treated data" and "Fused data".
    If the "Treated data" radio button is deselected while the "Fused data" radio button was selected, the "Fused data" radio button becomes inactive.

→ Once the outliers and suspicious arrays have been created for the fused data:
  – the "filterfused" pop-up menu is enabled,
  – the "OK" push-button controlling it is also enabled.

Instructions
In order to successfully process and fuse its data, the engineer must follow these instructions in order:

1 - Start by loading the data sets.
2 - Plot the desired sites to make sure the data is compatible and doesn’t present any visible mistake.
3 - Visualize the outliers and suspicious points, examine closely to decide if or how they must be filtered out.
4 - Filter and map appropriately the data by selecting the right options for both actions.
5 - Plot the treated data alongside the original. Make sure the differences are coherent.
6 - Fuse the data.
7 - Plot the fused data alongside the original one. Make sure the fusion present coherent results.
8 - Visualize the outliers and suspicious points for the fused data. Examine closely to decide what to do.
9 - Filter according to the previous observations.
10 - Possibly, plot the treated data along with filtered fused data.
These instructions and more are available to the user by simply typing "Help Fusion" in the Matlab command window:

**Figure 36:** Output of the "help Fusion" command.

**Limitations** As mentioned in the "Note" and "IMPORTANT" sections of the `help Fusion` command, the GUI has a few limitations:

- Due to the fact that the number of plots and figures is internal to the `myplot` and `myvisual` functions, no counter can be attached to them.
  
  In order to be able to have a "Close plots" button, that really comes in handy when more than 1 site is plotted — there are 6 figures per sites * number of plotted sites for the `myplot\_map` function, and 8 for `myvisual` function —, a counter was installed inside the GUI, counting 6*number of sites selected if the "Plot" button is clicked on, and 8 if the "Visualize outliers & suspicious" button is clicked on.
In this configuration, if the user manually closes even one figure, the count becomes then inexact and an error message is displayed if the "Close plots" button is pressed. However, the program does not crash and the user can proceed with the next step.

- The last filtering-mapping performed gives the input for the fusion. This is highly important. If the user chooses to filter the data, then filter it 2, the data that will be used for the fusion is the data filtered 2. The engineer must keep this in mind in order to have the correct data used for the fusion. This is not true for the outliers & suspicious visualizing function, where you can, for instance, first observe the bad points in the fused data, then the original data, and start the filtering of the fused data right after. The `myfilterfused` function will take the correct arrays for its filtering.

### III.4.3 Integration within CEASIOM

The GUI completed, the Fusion module worked as a stand-alone program. It needed now to be integrated as one of CEASIOM modules. The main CEASIOM interface is created by the AMB module. After launching it, the user can call to all the other modules in CEASIOM. In order to integrate my Fusion module to CEASIOM, a few modifications had to made in AMB.

- The first thing was to add a new sub menu to the AMB GUI. I called it ViewFilterMap Fusion, referring to the different operations that can be carried out within this module,

- The second was to code the appropriate call backs for this sub menu, so that the Fusion module be launched on click.

Once this is done, CEASIOM can be started and, when the fusion is needed, or potentially at any time the user simply might want to visualize its data, the Fusion module can be called and used. When the process is completed, the engineer closes the Fusion module and can proceed with the SDSA module, using the output saved in the Matlab Workspace or the .xml output from the Fusion.

56
IV Conclusion

The objective of this mission was to implement an additional module for the CEASIOM environment. The new module were to automatize the process of fusing the different aerodynamic tables, generated by CEASIOM using adaptive fidelity methods. It would also include a filter, a custom-made mapping to enhance the accuracy of the results generated by the fusion, as well as visualization tools to spot any abnormality in the data. It should also be managed by by a Matlab GUI and integrated into CEASIOM.

Achievements

Five main functions were coded to meet the requirements of the scope of work. These functions all work perfectly, but the `myfiltermap` function. The mapping part of the function lacks in precision, and can thus cause more damage than good. The main function in this module, `myfusion`, gives very good results on test data. A run on actual data has however yet to be done to validate it fully. Finally, the GUI works very well. The engineer can call the module from a submenu in the CEASIOM interface, and from there, carry out all tasks relative to the fusion. The GUI was built so that any mistake or call to a function in the wrong order would be avoided, by rendering buttons active or inactive according to the status of the process. "Help" and "About" buttons are also available, as well as a detailed list of instructions and limitations of the program, by simply typing "help Fusion" in the Matlab command window, as for any Matlab function.

Future Work

→ Though the mapping part is deficient, it can be used as a good basis for improvement. A first measure would be to apply it on a unified set of data so that a distance coherent with the mapped data be maintained between all the values. The overall inaccuracy problem could certainly be solved by refining the mathematical formula used in the process, so that the stretch is realized with more precision.

→ The fusion works well but can still be improved. A way to do this would be the possibility of assigning weights to the data sets, so that the higher fidelity data sets weigh more than the lower fidelity ones in the interpolation calculations. The data set presenting the highest fidelity would then impose the trend for the values, thus improving the overall accuracy of the results. However, by its architecture, kriging can not be used or modified to that purpose. A way to realize this could be to use Radial Basis Functions.
Bibliography

Appendices
I Kungliga Tekniska Högskolan (KTH) - The Royal Institute of Technology[1]

I.1 General points

The Royal Institute of Technology in Stockholm was founded in 1827. It officially became a school of higher education in 1927 when allowed to deliver the degree of Doctor of Technology. It includes courses and research programs in various fields, from all the branches of engineering, to natural sciences, architecture, urban planning or management. It also houses several competence centers, and research programs financed by various scientific foundations.

It is divided in 11 different «schools» by field of activity:

- the School of Architecture and the Built Environment
- the School of Biotechnology
- the KTH Business Liaison
- the School of Chemical Science and Engineering
- the School of Computer Science and Communication
- the School of Electrical Engineering
- the School of Information and Communication Technology
- the School of Industrial Engineering and Management
- the School of Engineering Sciences
- the School of Technology and Health
- the Scientific Information and Learning

KTH main campus has been located in central Stockholm since 1917, but several of its schools are found in Stockholm’s periphery: Kista for the School of Information and Communication Technology, or Flemingsberg for the School of Technology and Health.

KTH is also part of a vast network of partnership with other universities, in Sweden as well as in all parts of the world: Europe, Australia, America and Asia.

I.2 Key numbers

KTH is a very active and world community involved university on several accounts, with about just over 13,000 full-year equivalent undergraduate students, more than 1,500 active postgraduate students and 2,935 full time equivalent employees.

**Academic facts** In terms of education, KTH offers 15 architecture and engineering programmes, 55 master programmes and 8 B.Sc engineering course. About 13,344 student (30% women) attend the classes each year, and there are more than 1,500 doctoral students (29% women). In 2009, KTH delivered entre autres 955 Masters in Architecture and Engineering and 222 PhD.

KTH also has responsibility for 18 national research centers, such as Vinnexcellence, Linné, SSF, Mistra and STEM Centers.
International KTH has developed extensive international relations. During 2008, KTH welcomed approximately 1500 new international master’s students and 1153 exchange students. It is part of not less than 11 prestigious academic networks with nations all around the world: BALT-TECH, CESAER, CLUSTER, ECATA, NORDTEK, Pegasus, T.I.M.E with Europe, GE4 with South America, Magellæas with China or WGLN2 with the USA. It is also part of the ERASMUS MUNDUS, LIFELONG LEARNING ERASMUS, LEONARDO DA VINCI, EU MoU, SMARTIE, MFS, NORDPLUS, TEMPUS and Asia-Link, all important international projects that aim to facilitate and improve students and teachers’ mobility as well as enhancing the quality of higher education. Some of these projects, such as the MFS, LINNAEUS, TEMPUS or Asia-Link, specifically aim at developing countries, in order to increase knowledge about them, as well as creating and strengthening scientific exchange between them and, through Sweden, Europe.

Fundings For its researches and education programs, KTH has fundings for a total of 322 million euros per year, essentially coming from external financings - of which private sources (49%), governing agencies (34%), and the Swedish Research Council (21%). The rest comes in form of subventions from the EU Framwork programs and Vinnova, the Swedish Agency for Innovation Systems (respectively 15% et 13%)

I.3 Miscellaneous

KTH’s history counts several interesting facts. Sweden’s first reactor was installed in a cellar in the main campus area in the early 50’s, and in the same street, the very first TV station started to emit. KTH can also be proud of having counted among its teachers a Nobel Prize winner. Hannes Alfvén, who had been professor of Electromagnetic Theory and Measurement Technology at KTH since 1940, shared the Nobel Prize for physics in 1970, as the citation read, “for fundamental work and discoveries in magnetohydrodynamics with fruitful applications in different parts of plasma physics.”[8] He continued teaching in KTH till his retirement, year 1991. The asteroid 1778 Alfvén was named in his honor.

I.4 The Aerodynamics division

This mission took place within the Aerodynamics division of the KTH department of Aeronautical and Vehicle Engineering.

I.4.1 KTH Aeronautical and Vehicle Engineering department [4]

KTH Aeronautical and Vehicle Engineering is part of the School for engineering sciences. It concerns itself with the science and research behind air, rail, ground, and sea vehicles. The department is divided in 7 research areas, each focusing on a different aspect of vehicle engineering:

- Aerodynamics
- Flight Dynamics
- Lightweight Structures
- MWL - Marcus Wallenberg Laboratory for sound and vibration research
- Naval Systems
- Rail Vehicles
- Vehicle Dynamics
It possesses extensive experimental resources, such as a L2000 wind tunnel, in order to validate the theoretical results obtained by the researchers. The department also has responsibility for the VINNX Centre of Excellence for ECO2 Vehicle design and coordinates the Gröna Täget research program. KTH and undergraduate teaching provide for about 45% of the department total fundings, the rest coming from external contracts. The department has a vast network of partners at national and international level, including government agencies and academic institutions.

1.4.2 Activities[5]

The researches carried out in this division focus on the study of the present aerodynamic limitations, as well as investigate new aeronautical concepts, by using, enhancing or creating computational tools, hence applied or experimental CFD. The possible applications are vast, and include, for instance, providing computer-based design studies, thus limiting the need for costly experiments.

The division has been involved in several european funded projects:

- the Rear fuselage and EMpennage Flow Investigation (REMFI) project, that investigated the complex physics of the flow in order to come up with innovative and more efficient tail designs,
- the HISAC project, aiming at finding an environmentally friendlier high speed aircraft, that involved 37 partners in 13 different countries (ended end of 2009),
- the VFE-2,
- the SimSac project, that will be developed later in this paper.

The department is also involved in the development of two softwares: Tornado and LIC.

- LIC, for Line Integral Convolution is a method providing a visualization of the flow patterns in a similar way as the ones that can be observed in wind tunnel testing in wind-blown sand or oil-flow procedures.
- Tornado is a wing design code developed within a collaboration between KTH, the University of Bristol, the Linköping University and the Redhammer Consulting Ltd. It allows the user to obtain quasi-immediate feedback on changes in design in terms of the aircraft’s performances, thus providing the engineers with useful information early in the design process. Constantly developed and updated, Tornado is used in universities and companies around the world and has become a prominent software in the world of aircraft design.

The researchers in the department have also published several papers and books of scientific values in the past years.
I.4.3 Organisation

![Organization chart of the CFD division within the KTH Aeronautical and Vehicle Engineering department]

Figure 37: Organization chart of the CFD division within the KTH Aeronautical and Vehicle Engineering department