Development of a Gust Generator for a Low Speed Wind Tunnel

Jan Andreas Geertsen

Master Mechanical Engineering

Master Thesis FS 2020

Institute of Fluid Dynamics
ETH Zürich

conducted at

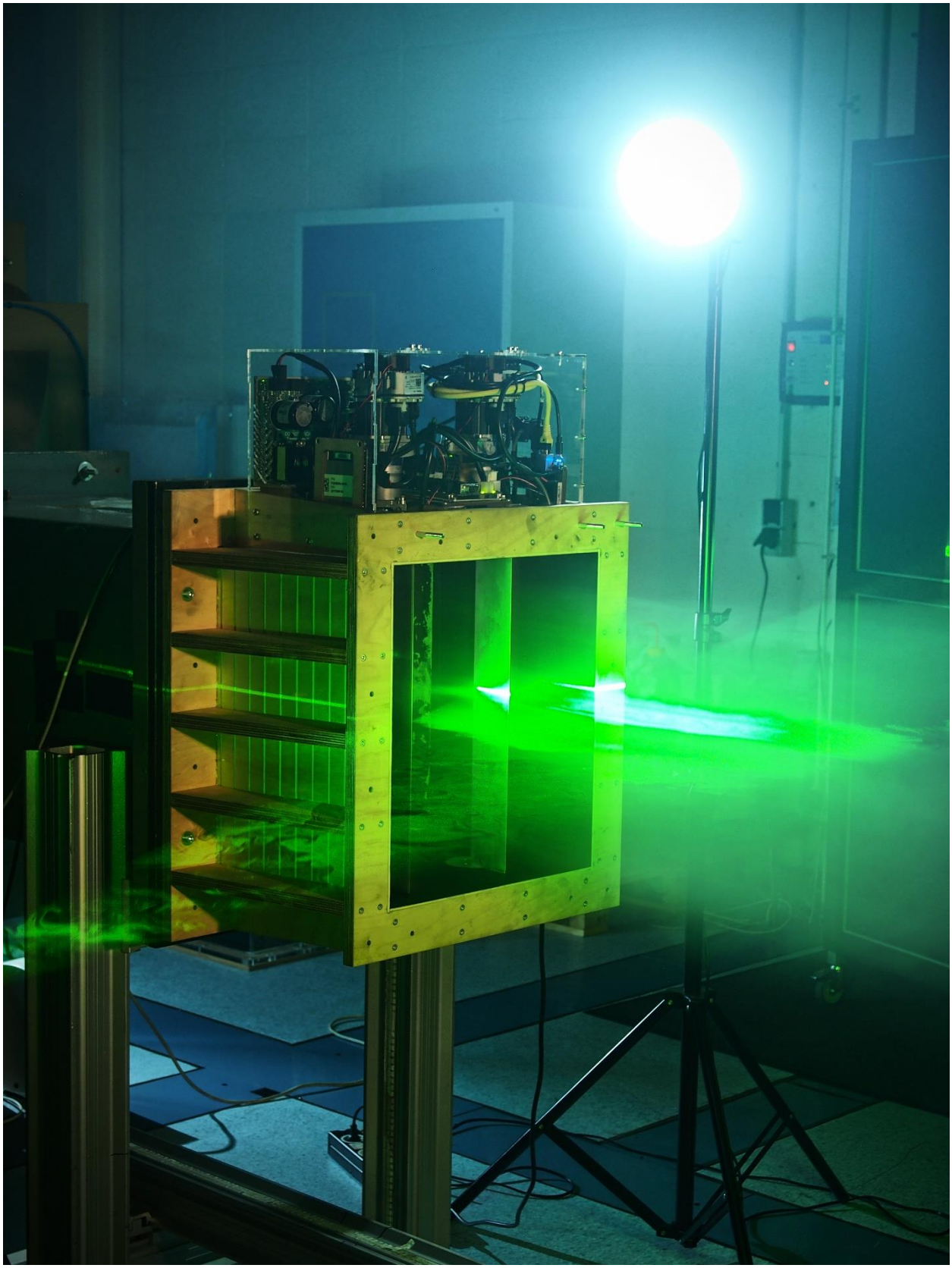
TU Delft
Faculty of Aerospace Engineering
Delft, Netherlands

Supervisor: Dr. J. Sodja, TU Delft
Dr.ir. R. De Breuker, TU Delft

Professor: T. Rösgen

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TU Delft Delft University of Technology



Abstract

Gust load is considered to be one of the most demanding load cases for an aircraft. In the strive for better fuel efficiency modern aircraft design is heading towards wings of increasing aspect ratio as well as more lightweight structures, making them more sensitive to gusts. It is therefore apparent that gusts and the structural response to it are an important topic of research in academia as well as in the aerospace industry.

Numerical methods to simulate fluid dynamics, structural mechanics and the interaction between the two have improved rapidly over the past years and are widely used in research to validate theoretical concepts, to improve the understanding of various phenomenon or to optimise initial designs. Experimental means are however as important as ever to validate theoretical as well as simulated results.

The most commonly used equipment for aerodynamical experiments is hereby the wind tunnel which is by itself however not capable of generating gusts. As a result, a gust generator is needed which modulates the airflow and generates gusts.

The present work describes the development of such a gust generator for a specific low speed wind tunnel at Delft University of Technology.

A preliminary design study was performed to identify requirements as well as the restrictions given by the designated wind tunnel. An initial concept was derived. The overall geometry of the system was optimised by means of computational fluid dynamics with regard to a gust as uniform as possible and a gust velocity as high as possible. The optimised geometry was used to develop the final design.

The gust generator was successfully manufactured, and software was developed to control the gust generator. The final prototype could be realised as a fully enclosed system only needing an external computer to provide the necessary input parameter resulting in an easy to use piece of equipment.

The final prototype was tested, and it could be proven that the gust generator is capable of producing the desired gusts, however the final test results were inconclusive regarding the gust uniformity as well as the time resolved gust shape. Further testing is therefore required.

Zusammenfassung

Böen müssen bei der Auslegung eines Flugzeuges berücksichtigt werden und stellen einen anspruchsvollen Lastfall dar. Im konstanten Streben nach immer höherer Effizienz und geringerem Treibstoffverbrauch ist ein Trend hin zu Tragflächen mit zunehmender Streckung sowie zu generell immer leichteren Strukturen erkennbar. Dies Entwicklung bringt aber eine höhere Empfindlichkeit gengenüber Böen mit sich. Es ist daher offensichtlich, dass Böen und die strukturmechanische Antwort darauf ein wichtiges Thema der Forschung sowohl im akademischen Umfeld als auch in der Luftfahrtindustrie sind.

Numerische Methoden zur Simulation von Strömung, der Strukturmechanik und der Wechselwirkung zwischen den selbigen haben sich in den letzten Jahren kontinuierlich verbessert und werden in der Forschung verbreitet eingesetzt um theoretische Konzepte zu validieren, um verschiedener Phänomene besser zu verstehen oder Entwürfe zu optimieren. Experimentelle Methoden bleiben jedoch unverändert wichtig, um sowohl theoretische als auch simulierte Ergebnisse zu validieren.

Üblicherweise wird für aerodynamische Experimente auf einen Windkanal zurückgegriffen. Dieser ist jedoch übelicherweise nicht in der Lage, Böen zu erzeugen. Folglich wird ein Böen Generator benötigt, welcher den konstanten Luftstrom kontrolliert beeinflusst und somit Böen erzeugt.

Die vorliegende Arbeit beschreibt die Entwicklung eines solchen Böen Generators für einen spezifischen Windkanal im niedrigen Geschwindigkeitsbereich an der Technischen Universität Delft.

Es wurde eine Designstudie durchgeführt, um sowohl die Anforderungen als auch die durch den vorgesehenen Windkanal gegebenen Randbedingungen zu ermitteln. Ein Konzept wurde erarbeitet. Die Geometrie des Systems wurde mittels numerischer Strömungssimulationen im Hinblick auf eine möglichst gleichmäßige Böe bei gleichzeitig hoher maximaler Böen Geschwindigkeit optimiert. Die finale Konstruktion wurde dann basierend auf der Optimierung erstellt.

Ein Prototyp wurde erfolgreich gefertigt und Software zur Steuerung des Böen Generators wurde entwickelt. Der Böen Generator konnte als in sich geschlossenes System realisiert werden, welches lediglich einen Laptop zur Eingabe der notwendigen Steuerungsparameter benötigt. Somit wurde ein benutzerfreundliches und einfach zu bedienendes System entwickelt.

Der Böen Generator wurde getestet, und es konnte nachgewiesen werden, dass er in der Lage ist, die gewünschten Böen zu erzeugen. Die Testergebnisse waren jedoch hinsichtlich der Gleichmäßigkeit der Böen und dem zeitlichen Verlauf der Böen nicht ausreichend aussagekräftig. Weitere Tests sind daher erforderlich.

Acknowledgments

Never would I have anticipated that I will write my master thesis during a global health crisis. The measures taken to fight the global COVID-19 outbreak affected every aspect of the daily life all around the world. Given this situation, I consider myself lucky, that I could finish this thesis as planned. I would like to thank everybody who supported me and made it possible that I could continue my work.

In particular, I would like to express my gratitude to my supervisor Dr. Jurij Sodja for his supervision of this thesis. His support was always very helpful and much appreciated. His assistance throughout the whole thesis not only on academic matters but also with administrational problems made this thesis possible.

I would also like to thank Dr.ir. Roeland De Breuker for giving me the unique opportunity to write my thesis at the Faculty of Aerospace Engineering at Delft University of Technology and my master tutor at ETH Zürich, Prof. Dr. Thomas Rösgen, for supporting my aspiration to write my master thesis externally at TU Delft.

Furthermore, I would like to thank the technical staff for helping me with all practical matters. Special thanks goes herby to Peter den Dulk for manufacturing my parts with excellent precision, to Fred Bosch and Johan Boender for their straightforward support during the assembly of the gust generator and to Nico van Beek, Peter Duyndam and Dennis Bruikman for their support during the wind tunnel experiments.

I would also like to thank Dr. Andrea Sciacchitano for helping me with the set-up of the PIV system.

Additionally, I would like to express my gratitude to my room mates in Den Haag and Delft as well as to my family back home. The former for enduring my complaining and for giving me company when social interaction was otherwise scarce. The later for having me back home at the peak of the crisis in spring even though I said that I moved out for good two years ago.

Finally, I would like to thank my girlfriend for her love and support even over a distance of hundreds of kilometres. Thank you for motivating me after a bad day, thank you for proof reading page after page and thank you for simply always being there for me.

Development of the longitudinal and axial gust generator for a wind tunnel

Supervisor: Jurij Sodja and Roeland De Breuker
Location: ASCM, TU Delft
Starting date: asap

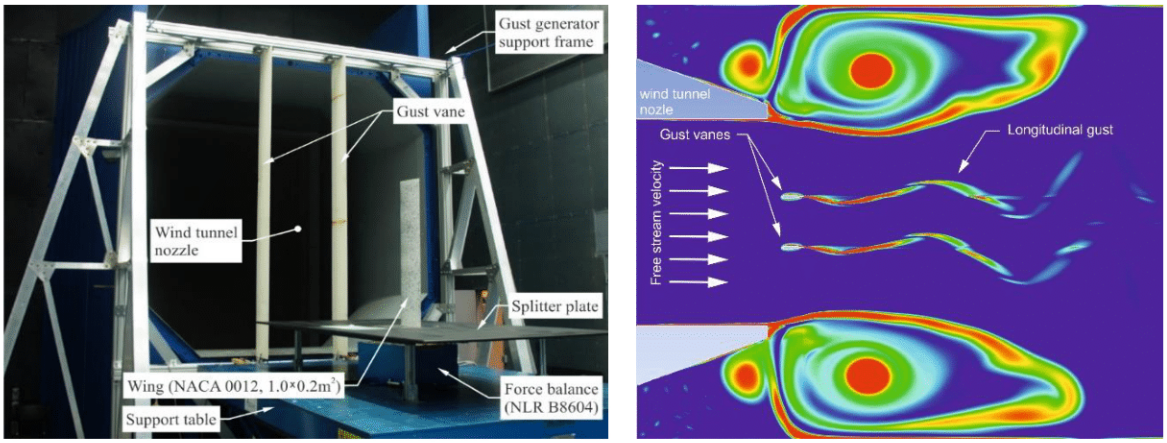
Project description:

Gust response and dynamic aeroelastic phenomena are becoming increasingly important for modern lightweight aerospace structures due to a constant drive for weight reduction making them increasingly susceptible to aeroelastic effects. To be able to investigate such aeroelastic structures experimentally in a wind tunnel environment, some means of aerodynamic excitation is required. Such excitation is usually provided by a gust generator. In a conventional manner, only longitudinal gusts are created by deflecting the airflow in the lateral direction by a set of gust wanes as shown in the bottom picture.

The thesis will focus on developing a gust generator that will be able to produce both longitudinal as well as axial gusts (wind shear). The gust generator will be built in a dedicated test section and installed in the M tunnel at TU Delft in order to test a lateral and axial gust response of the aeroelastic apparatus that is readily available.

Project outline:

1.) Review of the state-of-the-art and conceptual design of the gust generator
2.) Detailed design using CFD tools
3.) Mechanism and control system development
4.) Manufacturing of the gust generator
5.) Experimental characterisation of the gust generator using PIV



a.) Gust generator setup in the OJF wind tunnel

b.) CFD simulation of the gust generator perfomance

Figure 1: Gust generator setup

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Eigenständigkeitserklärung

Die unterzeichnete Eigenständigkeitserklärung ist Bestandteil jeder während des Studiums verfassten Semester-, Bachelor- und Master-Arbeit oder anderen Abschlussarbeit (auch der jeweils elektronischen Version).

Die Dozentinnen und Dozenten können auch für andere bei ihnen verfasste schriftliche Arbeiten eine Eigenständigkeitserklärung verlangen.

Ich bestätige, die vorliegende Arbeit selbständig und in eigenen Worten verfasst zu haben. Davon ausgenommen sind sprachliche und inhaltliche Korrekturvorschläge durch die Betreuer und Betreuerinnen der Arbeit.

Titel der Arbeit (in Druckschrift):

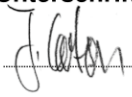
Developement of a Gust Generator for a Low Speed Wind Tunnel

Verfasst von (in Druckschrift):
Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich.

Name(n):	**Vorname(n):**
Geertsen	Jan Andreas

Ich bestätige mit meiner Unterschrift:
- Ich habe keine im Merkblatt „Zitier-Knigge“ beschriebene Form des Plagiats begangen.
- Ich habe alle Methoden, Daten und Arbeitsabläufe wahrheitsgetreu dokumentiert.
- Ich habe keine Daten manipuliert.
- Ich habe alle Personen erwähnt, welche die Arbeit wesentlich unterstützt haben.

Ich nehme zur Kenntnis, dass die Arbeit mit elektronischen Hilfsmitteln auf Plagiate überprüft werden kann.

Ort, Datum	**Unterschrift(en)**
Delft Niederlande, 15.08.2020	

Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich. Durch die Unterschriften bürgen sie gemeinsam für den gesamten Inhalt dieser schriftlichen Arbeit.

Contents

List of Figures

List of Tables

Abbreviations

CFD	Computational Fluid Dynamics
OJF	Open Jet Facility
RSC	Rotating Slotted Cylinder
GV	Gust Vane
GVU	Upper Gust Vane
GVL	Lower Gust Vane
TW	Test Wing
WT	Wind Tunnel
FSI	Fluid-Structure-Interaction
CAE	Computer Aided Engineering
AoA	Angle of Attack
SBC	Single Board Computer
PWM	Pulse Width Modulation
PIV	Particle Image Velocimetry
PRU	Programmable Real-Time Unit

List of Symbols

t	Time or thickness
f	Frequency
k	Reduced frequency
c	Chordlength
l	Length
φ	Angle
T	Torsional moment
G	Shear modulus
μ	dynamic viscosity
ρ	density
Cl	Coefficient of lift
Cd	Coefficient of drag
L	Lift
ω	Angular velocity
$\dot{\omega}$	Angular acceleration
v_{gust}	Gust velocity
v_G	Maximum gust velocity of 1-cos gust
$\bar{C}$	Mean aerodynamic chord
s	Distance travelled by aircraft in gust or wingspan
v_{max}	Maximum flow speed
v_y	Velocity in y-direction, gust velocity
v_x	Velocity in x-direction, flow speed
α_{gust}	gust angle
v_{ymax}	Maximum gust velocity measured in simulation
$v_{ymaxnorm}$	normalised maximum gust velocity measured in simulation
$d_{\overline{vymax}}$	Deviation of velocity across gust generator

$d_{\overline{vymaxnorm}}$	normalised eviation of Velocity across gust generator
opt	Optimisation parameter
w_1	Weight for maximum gust velocity
w_2	Weight for gust velocity deviation
u_0	Free stream velocity
δ	Boundary layer thickness
y^+	Distance from the wall in wall units
${y^+}_p$	First cell height in wall units
y_p	First cell height
y_h	First cell height as defined by Fluent
u_τ	Friction velocity
τ_w	Wall shear stress
K_s	Sand grain roughness
R_a	Mean roughness
${K_s}^+$	Non-dimensional roughness height
u^*	Non-dimensional velocity (alternative to u_τ)
$y_{interest}$	y-coordinate range of interest in test section
Δy_{test_wing}	Thickness of test wing
I_T	Torsion constant
s_{wall}	Wall thickness
K	Torsional stiffness
M_{aero}	Aerodynamical moment
φ_{max}	Maximum gust vane angle
$\lvert\omega_{max}\rvert$	Absolute maximum gustvane angular velocity
$\lvert\dot{\omega}_{max}\rvert$	Absolute maximum gust vane angular acceleration
$\dot{\omega}_{maxM}$	Maximum angual acceleration at actuator
$\dot{\omega}_{maxL}$	Maximum angular acceleration at load side
T_{mech}	Mechanical torque

$T_{maxmech}$	Maximum mechanical torque
$T_{maxload}$	Maximum total torque
R	Inertia Ratio
I_{GV}	Moment of inertia of the gust vane
I_M	Moment of inertia of the mounting parts
I_G	Moment of inertia of the gearbox
I_A	Moment of inertia of the actuator
GR	Gearing ratio
Δt	time step size of one motion command
t_l	lag time due to motion command validation
$\dot{\omega}_{inew}$	adapted angular acceleration due to lag for motion command
$\dot{\omega}_i$	original angular acceleration for motion command
ω_i	original angular velocity for motion command
ω_{inew}	adapted angular velocity due to lag for motion command

1 Introduction

Everyone who boarded an aircraft at least once in his life has most likely witnessed turbulence or an air pocket during their flight. Turbulence, in the popular sense, can be described as a continuous series of gusts, whereas a single down wards gust is commonly referred to as an air pocket. The load cases associated with such phenomenon are considered to be among the most severe in aircraft design [1]. As a consequence, gust loads are a part of the certification process to ensure the air worthiness of newly developed aircrafts (see fig. 1) [2]. Recent development in aircraft design is furthermore heading towards increasing wing aspect ratio as well as more lightweight structures in general, increasing their sensitivity to gust loads.

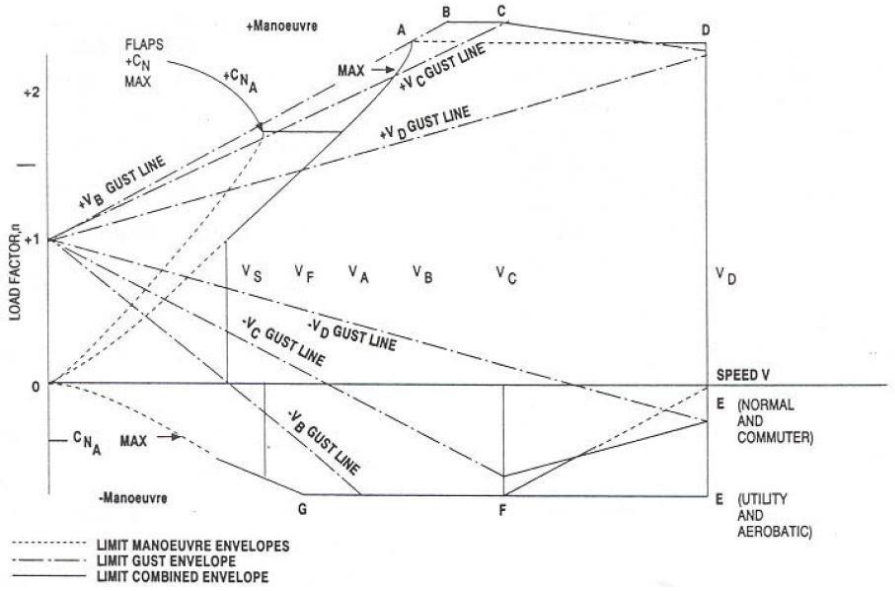


Figure 1: Gust envelope [2]

It is therefore apparent that gusts and any structural response related to it is a topic of high interest in any aerospace development endeavour as well as in scientific research in the field. In the past, numerous models have been proposed to address this topic theoretically as well as with means of computational fluid dynamics (CFD) [1], [3]. Any results generated by theoretical means as well as by simulation do however require experimental validation. The most widely used piece of equipment to generate such experimental results in the field of aerospace engineering can be considered to be the wind tunnel, which is typically only capable of producing an airflow in a fixed direction and of constant or slowly changing flow speeds. Subsequently, a device is required to modulate the constant flow in a controllable manner to generate repeatable gusts. Such a device is referred to as a gust generator. Multiple concepts have been proposed and realised in the past, of which a selection is described in section 2. The present work is based on a gust generator developed and built at TU Delft for a large open jet wind tunnel called the Open Jet Facility (OJF) (see fig. 2) [4]. This wind tunnel features a cross section of 2.85 x 2.85 m and is capable of flow speeds of up to 35 m/s.

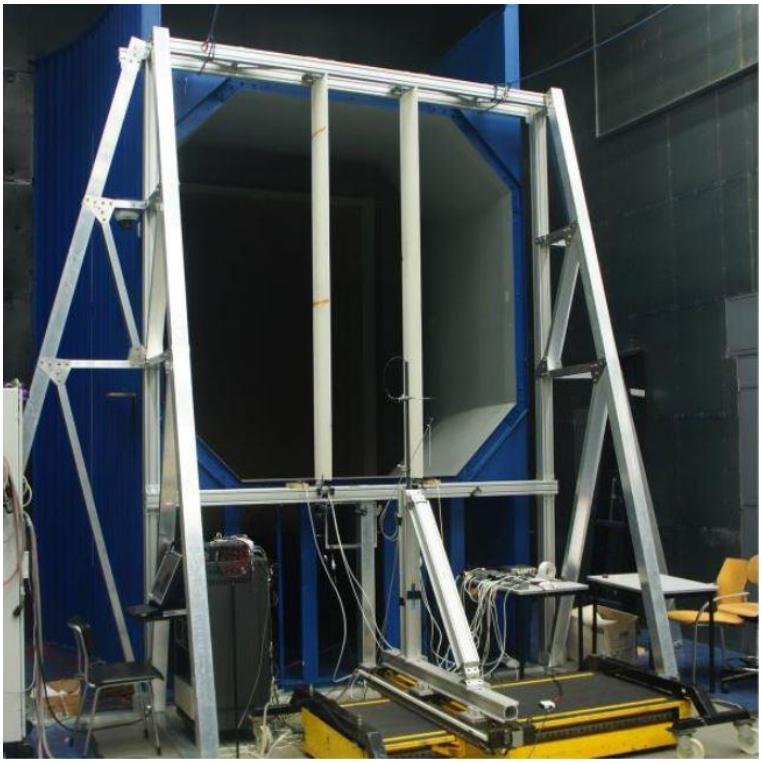


Figure 2: Gust generator for OJF [4]

This piece of equipment performs well and is used on a regular basis in current research [5]. However, it became clear that it is disproportionate to use such large-scale equipment for small-scale experiments. Thus, the desire to have a small enclosed system for one of the smaller low speed wind tunnels available at TU Delft arose. The development and realisation of this system was the target of this thesis and is described in detail in the following sections.

2 State of the Art

The following chapter is intended to provide an overview over existing gust generators and their operating principle. Literature lists multiple existing or previously existing systems at various institutes around the world. The list below summarises some of the facilities developed and installed in the past 50 years. The list does not claim completeness:

Research Institute/University	**Year**	v_{max}	**Wind tunnel cross section**
NASA (USA) (Reed 1981)	1966	Mach 1.2	Square 4.9×4.9 m2
MIT (USA) (Ham, Bauer *et al.* 1974)	1974	37 m/s	Elliptical 2.13×3.32 m2
Duke University (USA) (Tang, Cizmas *et al*. 1996)	1996	25 m/s	Rectangular 0.7×0.53 m2
Virginia Tech (USA) (Grissom and Devenport 2004)	2004	15 m/s	Square 2.15×2.15 m2
TSAGI (Russia) (Kuzmina, Ishmuratov *et al*. 2005)	2005	30 m/s	Elliptical 4.0×2.33 m2
TSAGI (Russia) (Kuzmina, Ishmuratov *et al*. 2005)	2005	120 m/s	Circular 7 m diameter
University of Maryland (USA) (Koushik and Schmitz 2007)	2008	N/A	N/A
Politecnico di Milano (Italy) (Ricci and Scotti 2008)	2008	30 m/s	Rectangular 1.0×1.5 m2
University of Colorado (USA) (Roadman and Mohseni 2009)	2009	20 m/s	Square 0.34×0.34 m2
DLR (Germany) (Neumann and Mai 2013)	2010	Mach 0.75	Square 1.0×1.0 m2
ONERA (France) (Lepage, Amosse *et al.* 2015)	2011	Mach 0.73	Rectangular 0.76×0.8 m2
Beihang University (China) (Wu, Chen *et al*. 2013)	2012	24 m/s	Square 3×3 m2
Cranfield University (England) (Saddington, Finnis *et al*. 2014)	2015	14.5 m/s	Elliptical 1.52×1.14 m2
ARA (England) (Allen and Quinn 2015)	2015	Mach 0.85	Rectangular 2.74×2.44 m2
Politecnico di Milano (Italy) (Ricci, Adden *et al*. 2015)	2016	55 m/s	Rectangular 4.0×3.84 m2
Mitsui engineering (Japan) (San technologies website)	N/A	20 m/s	N/A
JAXA (Japan) (Kenichi, Shunsuke *et al*. 2015)	N/A	Transonic	N/A

Table 1: Overview of worldwide existing gust generator installations. If v_{max} of the gust generator was not available, then the value for the wind tunnel itself is listed [4]

The development of these systems is primarily driven by the experimental requirements as well as by the wind tunnel facilities they are intended to be used with. A selection of three individual gust generators and their working principle are described in more detail below. These three concepts were chosen as they represent fundamentally different concepts. Note that other ways of generating a gust have been proposed and realised.

2.1 Oscillating Airfoils

Probably the most common concept relies on oscillating airfoils. Configurations with only one as well as with two or more airfoils have been realised [4]. In the case of the gust generator developed by *Lancelot et al.* (2016), two airfoils are mounted vertically and can be periodically pitched resulting in a deflected air flow and subsequently a gust (see fig. 2). This type of gust generator needs more torque than others, but it is able to generate gusts in accordance with certification requirements [2]. This working principle allows for high flexibility with regard to executable motion profiles if each involved airfoil is individually controllable.

2.2 Rotating Slotted Cylinder

A design proposed by Tang *et al.* (1996) and built at Duke University, USA, makes use of a rotating slotted cylinder (RSC) behind the trailing edge of an airfoil [6]. The cylinder deflects the flow behind the airfoil and thus generates a periodic gust (see fig. 3). One complete gust cycle is hereby generated with every 180° of rotation of the cylinder.

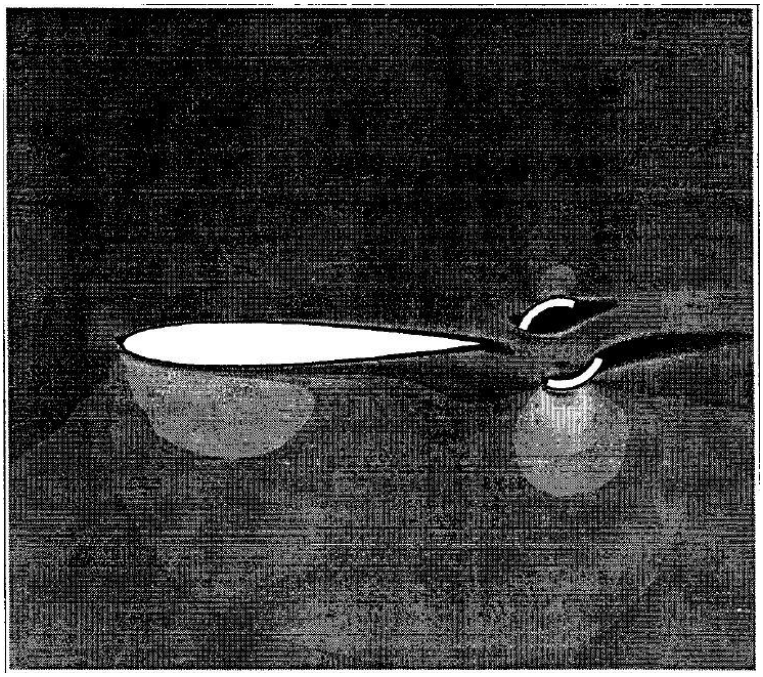


Figure 3: RSC concept [6]

This working principle has the advantage that it is mechanically simple and can be controlled very easily. In addition, the required torque is comparably low. Its application as a flutter exciter for flight testing has been suggested [7]. However, this concept allows for lower gust velocities compared to a concept based on oscillating airfoils. It is less flexible as well, as the only adjustable parameter of the generator itself is the rotational speed.

2.3 Active Turbulence Grid

A gust generator built at the University of Colorado Boulder, USA, by *Roadman et al.* (2009) used an active grid to generate turbulence. Herby multiple rows and columns of rhomboidal wings are mounted on shafts. These shafts are individually actuated and herby opening and closing the grid by rotating the wings (see fig. 4). By doing so vortices of different length scales shed off of the wings or other parts of the grid itself, which introduces turbulence of different length scales simultaneously [8], [9]

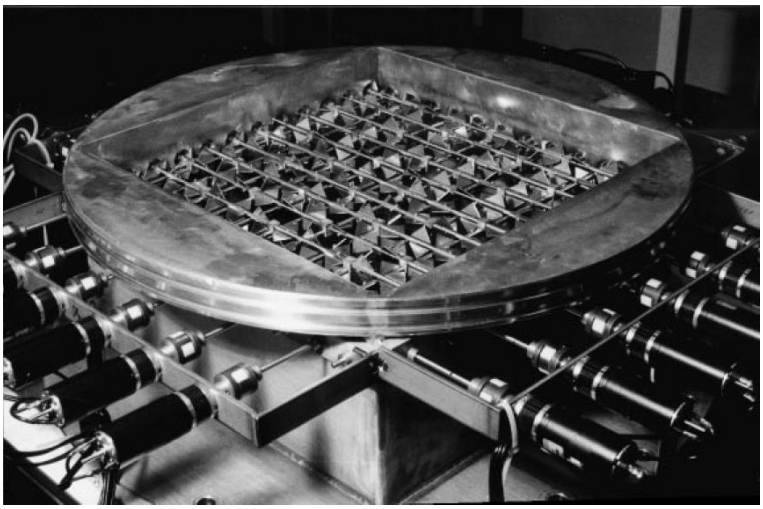


Figure 4: Active turbulence grid [9]

The goal of this concept is somewhat different of the other two, as it is not capable of generating uniform discrete gusts but is rather used to introduce continuous turbulence.

In summary it can be stated that multiple concepts exist, each with unique properties. As earlier stated, it is therefore necessary to choose a concept tailored to the intended use.

3 Problem Assessment

A problem assessment was performed to generate an understanding of the task at hand. The goal of this assessment was to determine the requirements the system must full fill, the boundary conditions in which it will have to operate and to generate a concept of the system yet to be designed. Additionally, a simplified estimation of the expected system stiffness was performed to determine how the system must be simulated in subsequent stages of the development process.

3.1 Problem Definition and System Requirements

The gust generator which had to be developed was required to generate two types of gusts. Both gust types are derived based on the gust described in the CS23 certification from the European Aviation Safety Agency [2]:

$$v_{gust} = \frac{v_G}{2}\left(1 - \cos\frac{2\pi s}{25\bar{C}}\right) \quad (3.1)$$

Hereby, v_G denotes the maximum gust amplitude, s describes the distance the airplane travelled into the gust and $\bar{C}$ the mean aerodynamic chord. With

$$s = v_{ref} * t \quad (3.2)$$

and

$$f = \frac{v_{ref}}{25\bar{C}} \quad (3.3)$$

where v_{ref} is the reference speed respectively the traveling speed of the aircraft and f the frequency of the gust, the formula can be simplified to

$$v_{gust} = \frac{v_G}{2}(1 - \cos 2\pi f t) \quad (3.4)$$

The gust described with equation 3.4 represents a single gust as seen in figure 5 and will be called “1-cos” gust in the present work. Directly derived from this type of gust is the continuous “sin” gust:

$$v_{gust} = v_G * \sin(2\pi f t) \quad (3.5)$$

The sin-gust represents a continuous gust which periodically changes direction. Note that v_{gust} in the case of a 1-cos gust ranges from 0 to v_G, where as it ranges from -−v_G to $+v_G$ for sin-gusts. Through out this work some of the results are described with regard to the gust angle:

$$\alpha_{gust} = \tan^{-1}\left(\frac{v_{gust}}{v_{ref}}\right) \quad (3.6)$$

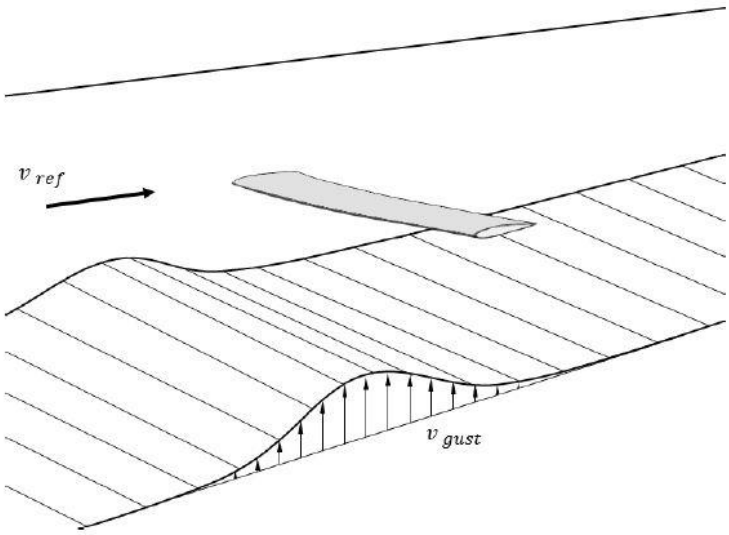


Figure 5: Representation of a 1-cos gust [4]

The gust generator needed to be designed based on equipment which already existed. This equipment highly constrained the design. A description of the equipment is given below:

- **Wind tunnel**
  The gust generator was intended to be operated with a specific wind tunnel at TU Delft, called W-Tunnel. The W-Tunnel is an open cycle wind tunnel. The W-Tunnel (see fig. 6) can produce flow speeds of up to 35m/s. To have a margin, a maximum flow speed of 30 m/s was considered as the maximum possible. The flow it generates is in general of a low turbulence intensity which can go as low as 0.5% under the right flow conditions. The cross section at the exit is 400 x 400 mm.

Figure 6: W-Tunnel at TU Delft

- **Test Section**
  The test section is designed to be used for aeroelastic experiments and consists of a rectangular tubular section made from acrylic glass into which a test wing of 160 mm chord length is mounted. The mounting of the test wing allows for pitching as well as for plunging motions. Additionally, the stiffness of the system related to each motion is adjustable [10]. The test section has a length of 515 mm and a cross section of 400 x 354 mm.
- **Adapter section**
  An Adapter section made from plywood is used to connect the test section to the wind tunnel. As the wind tunnel and the test section do not have the same cross section, the adapter narrows towards the test section. The gust generator was supposed to be built into this adapter section.

An overview of the geometrical situation is given in figure 7.

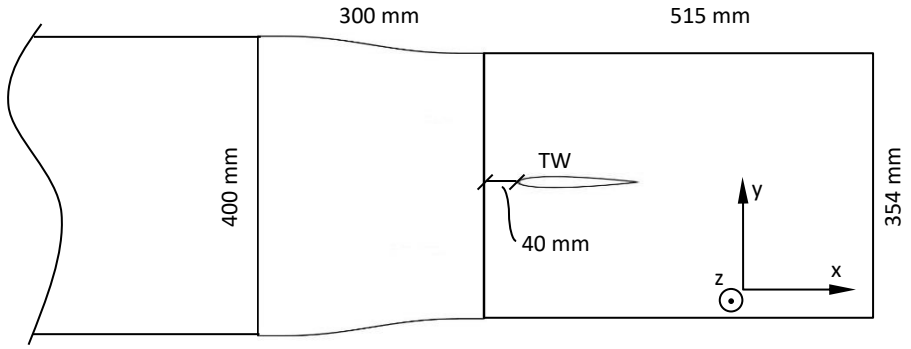


Figure 7: Geometrical situation

The gust generator should further be able to cover a range of reduced frequencies up to 0.2. The limit was set to 0.2 as anything above it is in the domain of highly unsteady aerodynamics. The reduced frequency is defined as follows:

$$k = \frac{2\pi f \frac{c}{2}}{v_{ref}} \tag{x}$$

where c is chord length. Considering the chord length of the test wing as well as the maximum flow velocity of 30 m/s, a maximum gust frequency of 12 Hz is calculated. All above mentioned requirements are summarized in table 2:

Requirement	Value
Gust type	1-cos, sin
Tolerable flow speed	30 m/s
Maximum gust frequency	12 Hz

Table 2: Physical Requirements

Additional to these physical requirements, a few user centred requirements were defined:

- The system should be enclosed and consist only of one device to facilitate the setup.
- The whole system control shall be done by logic components embedded in the system.
- Only a laptop or a computer without any further software shall be necessary to control the user input needed by the control software running on the system itself.

3.2 Concept

As described in section 2, multiple concepts for gust generators have already been developed. Given the limited space in the adapter section and the requirements defined in section 3.1 it was decided to use the same principle which has already proven to work with the gust generator built for the OJF at TU Delft. As a result, the gust generator consists of two identical airfoils (1), further referred to as gust vanes (GV), inside the adapter section (2). Each of them is individually driven by an actuator (3). A gearbox (4) is used to match the torque and the rotational speed of the actuator with the torque and rotational speed required to move the gust vanes.

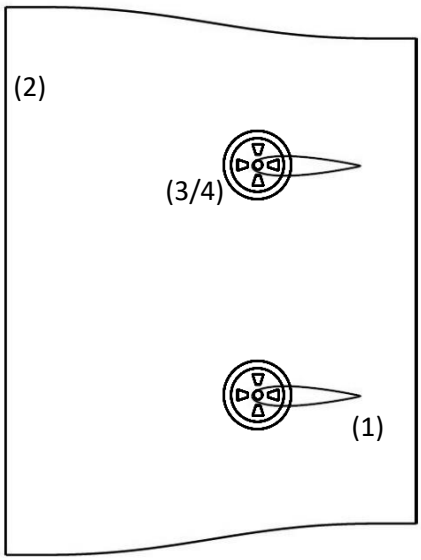


Figure 8: Chosen dust generator concept

The concept as described above offers several advantages:

- Mechanically simple
- Realisable in the limited space
- Allows to generate the desired gusts
- Each gust vane can be controlled individually which allows for synchronised as well as for asynchronous movements
- Existing know how due to previous development of similar gust generator at TU Delft

The profile of the gust vanes was chosen to be a NACA 0018 of 80 mm chord length as they were readily available as aluminium extrusions, facilitating the later construction process. A symmetrical 4-digit NACA profile in the range of NACA 0009 to NACA 0018 seemed to be a reasonable choice as they are dimension wise in proximity to the NACA 0012 profile, which is wildly used for aerodynamic simulations and experiments. In general a thicker profile leads to more wake turbulence but can handle higher angles of attack [11].

3.3 Rigidity Assessment

Additionally, a highly simplified estimation of the stiffness of the system was performed to assess if the system can be assumed to be rigid and subsequently one can refrain from performing a fluid structure interaction (FSI) study.

To this purpose three key components, being the adapter section, the gearbox and the gust vane, were considered. The adapter section (see fig. 9) in which the gust generator will be mounted was considered rigid enough without any further calculation, as it is a distinctively rugged design. It consists of an inner surface made from 4 mm plywood strengthened with a frame on the inlet as well as on the outlet. It is reinforced with 18 spars of plywood along the sides, which are all 18 mm strong.

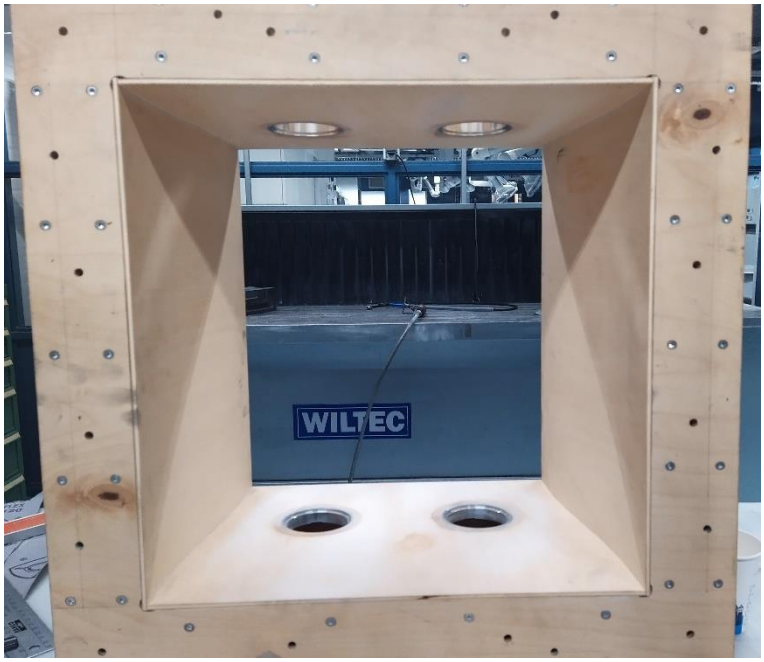


Figure 9: Adapter section, originally without holes

For the gearbox a realistically low backlash and high stiffness was defined as a requirement at this stage. A superficial study of available gearboxes in the necessary torque range lead to possible values of at least 0.5 Nm/arcmin for the stiffness and less than 15 arcmin for the back lash.

The stiffness of the gust vane was evaluated by only considering the rectangular middle section of the airfoil (see fig. 10) to calculate its torsion constant as

$$I_T = \frac{2tb^2a^2}{b+a} \qquad (3.7)$$

where t is the wall thickness and a and b are the lengths of the sides of the rectangle. This led to a value for I_T of 4.1*10^{-9} m^4, which is a very conservative estimation. Substituting the spread aerodynamic load with a single load acting at the centre of the gust vane, the torsional stiffness of the gust vane can be given as

$$K = \frac{T}{\varphi} = \frac{I_T G}{l} \qquad (3.8)$$

where G is the shear modulus of the material and l is half the length of the gust vane. This led to a stiffness value of 9.3 Nm/deg. Considering the stiffness of all three components it was therefore assumed that the complete system is rigid with regard to the expected loads and subsequently no FSI study was performed.

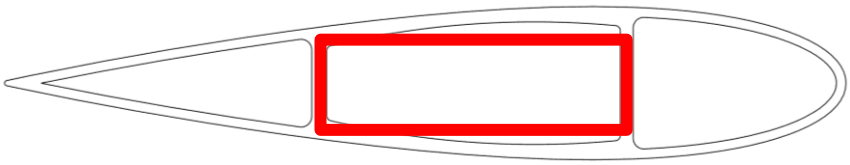


Figure 10: Gust vane cross section with simplified cross section for stiffness calculation

4 Fluid Simulations

The position of the gust vanes in relation to each other, as well as the distance between them and the point of interest in the downstream flow greatly affects the shape and strength of the measured gust. This could be shown during the development of the gust generator built for the OJF at TU Delft [4]. In contrast to this gust generator, the gust vanes will be placed inside a partially enclosed structure in the present case. Subsequently the gust vanes will be in the proximity of walls, which can heavily affect the air flow through the gust generator and must be taken into consideration. In consequence, a design optimisation process had to be done to define the optimal position of each gust vane. This optimisation was performed using CFD. The following sections describe the set-up procedure of the fluid simulation as well as intermediate results. The actual optimisation process and its results are described in detail in section 5.

A quick overview of the complete process involving fluid simulations is illustrated below:

1) Steady state simulations of the gust vanes were performed at one position to tune the simulation.
2) Transient simulation of the gust vanes at multiple positions and multiple frequencies were preformed to assess the coupling between the design parameters frequency and position.
3) Transient simulations of the gust vanes at multiple positions spread over whole design space were performed to generate the data for the optimisation.
4) The simulation data was postprocessed which included an interpolation to generate more data points.
5) All data points were evaluated and a weighted function was applied to find the optimal gust vane position.
6) The potentially interpolated data at said point was validated with a simulation.

4.1 Simulation Software

The computer-aided engineering packages (CAE) of Ansys Inc. were available to perform the fluid simulations needed for the optimisation procedure at hand. This package offers two solvers, CFX and Fluent, that can perform CFD related tasks. Both solvers are in theory able to perform the needed simulations. However, CFX is mostly known to be used for turbomachinery-related simulation. The decision between the two solvers was made on practical considerations. An incomplete overview of the differences is shown in table 3 [12]–[14].

Fluent	**CFX**
Offers overset meshing	Offers immersed body method
Offers mesh morphing	Offers mesh morphing
	User friendly post processing
Well documented user defined functions (UDF)	Overall beginner friendly
Capable of performing true 2D simulations	Not capable of performing true 2D simulations
	Well suited for turbomachinery simulations

Table 3: Fluent vs CFX

A decision was made to use Fluent due to its capability to handle overset meshes. This allows to mesh the wind tunnel and its attached sections independent of the gust vanes and the test wing. Therefore, a flexible model can be created which allows to optimize the design concerning the position of the gust

vanes, without excessive re-meshing for every design iteration. Additionally, it allows to build highly structured meshes. A second advantage is Fluent's capability of real 2D simulations which will save computational costs and therefore allows for finer meshing.

4.2 Environmental Conditions

The conditions for the simulation given by the geometrical appearance of already existing parts including the wind tunnel exit section and the basic fluid properties are described below. For the sake of completeness some conditions already established in section 3 are listed again.

- **Geometrical situation**
  The geometrical situation as partially described in section 3 can be seen below (fig. 11). Note that the position of the upper and lower gust vane (GVU and GVL) is not defined as their final position was the goal of the optimization. However, they were placed 120mm apart from each other (y -direction) and with their leading edge 300mm ahead of the test section (x - direction). The inlet section ahead of the gust generator has two lengths indicated, as simulations were run with both configurations. A detailed explanation for this can be found in section 4.3.1. A summary of all the important dimensions can be found in table 4. The reference point, $x = 0/y = 0$, for all further geometrical descriptions is defined to be on the centre line and on the exit of the gust generator/the entry of the test section, as indicated in figure 11.

Property	**Value**
Profile gust vane	NACA 0018
Profile test wing	NACA 0012
Chord length of gust vane	80 mm
Chord length of test wing	160 mm
Initial gust vane position, leading edge (x, y)	-300 mm, ±60 mm
Test wing position (x, y)	40 mm, 0 mm
Max gust vane angle	15°
Depth/height of wind tunnel (z – direction)	400 mm
Inlet length ahead of gust generator	450 mm/750 mm
Gust generator dimensions	300 x 400/354 mm (inlet/outlet)
Test section length	515 mm
Reynolds number:	
Wind tunnel, including the gust generator and the test section (WT)	2598000/3214100[1]
Gust vanes (GV)	164300
Test wing (TW)	328600

Table 4: Geometrical properties

[1] Only for completeness, cannot be considered to be exact, as the rest of the wind tunnel ahead of the inlet section is neglected

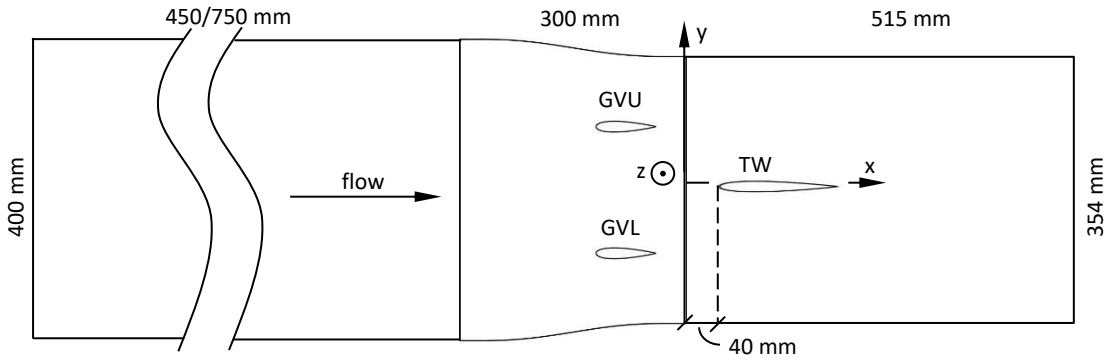


Figure 11: Geometrical situation as used for simulations

- **Fluid Properties**
  The fluid properties are given by the wind tunnel for which the gust generator is designed as well as by the surrounding environment. As the open circuit wind tunnel is situated in Delft, Netherlands, fluid properties were chosen according to the ICAO standard atmosphere at 0m MSL [15].

Property	Value
Medium	Air
Altitude	0 m MSL
Temperature	288.15 K / 15° C
Density	1.225 kg/m^2
Static pressure	101325 Pa
(Dynamic) viscosity	1.7894*10^{-5} kg/m/s

Table 5: Fluid Properties

4.3 Steady State Simulations

As an initial step of the optimisation procedure a steady state simulation was set up with the goal to establish and tune the fundamental components of the simulation such as the mesh and the solver. Additionally, first insights into the flow through the gust generator could be generated.

4.3.1 Mesh

The mesh was implemented as an overset mesh. This approach offers the advantage that different configurations as well as mesh movements can be performed without re-meshing. Consequently, some computational costs are saved as well as time which would be needed to manually adapt or change the mesh.

The overset mesh generated for the task at hand consists of four single meshes as seen in figure 12:

- Background mesh covering the whole enclosure consisting of the inlet, the gust generator, and the test section
- A mesh around each gust vane, both identical

- A mesh around the test wing

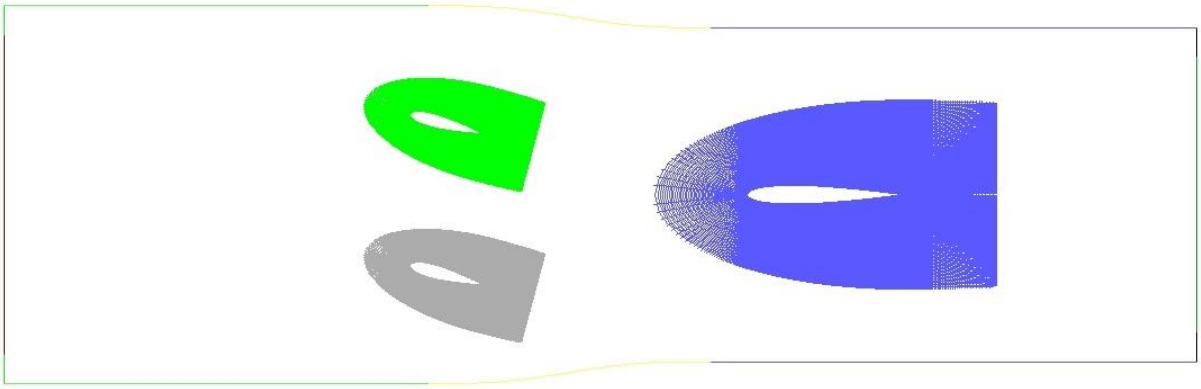


Figure 12: Complete mesh, background not shown

All four meshes were generated in a structured manner with all-quad elements. The meshes for the gust vanes as well as for the test wing were generated as a C-Type mesh (see fig. 13 and 14). The shape of the background, being a slightly deformed rectangle lead to an extremely simple mesh (see fig. 15). All meshes feature an additional zone dedicated to the boundary layer with gradually smaller cells towards the wall (see fig. 16) The boundary layer zone of the gust vanes and of the test wing are extended beyond their trailing edge to achieve higher accuracy in the wake zone of each airfoil (see fig. 17). Considering the rotational movement of the gust vanes, this refined wake zone fans out downstream. The same meshing scheme was used for the test wing mesh as well to facilitate the meshing process.

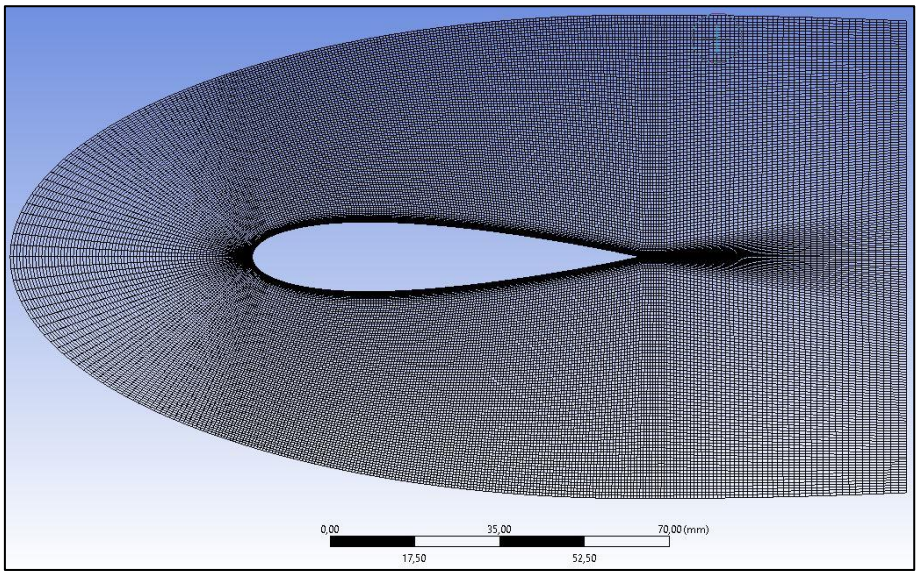


Figure 13: Gust vane mesh

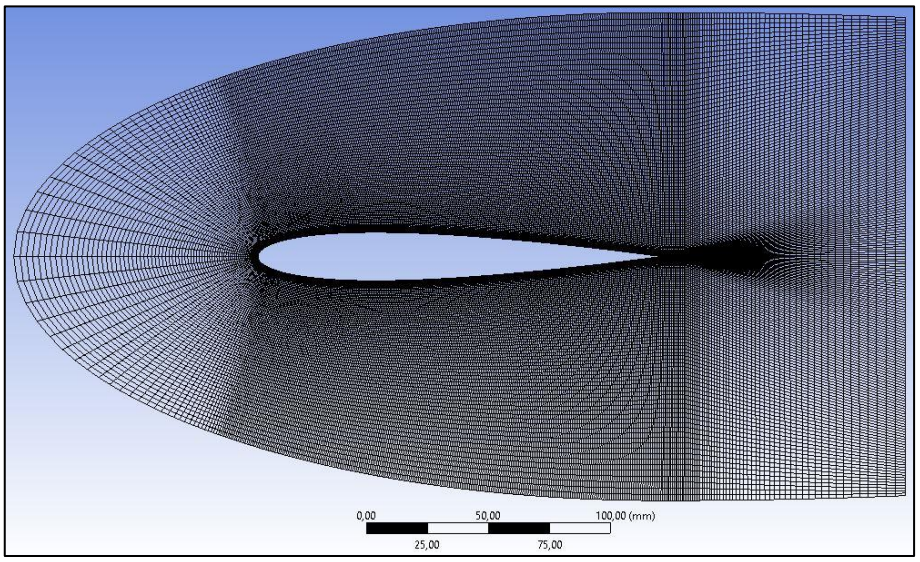


Figure 14: Test wing mesh

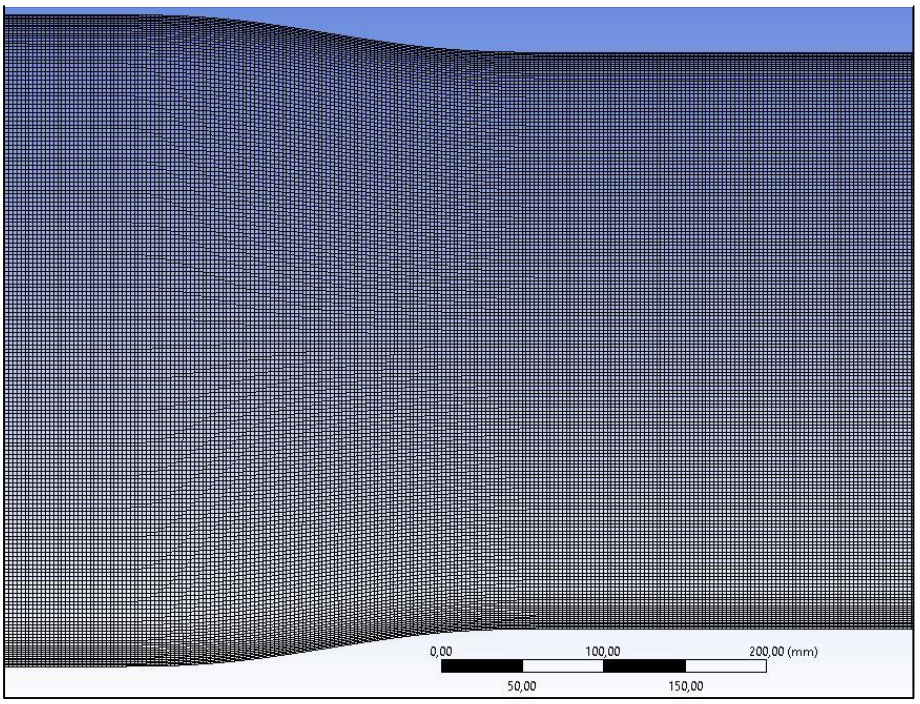


Figure 15: Section of the background mesh

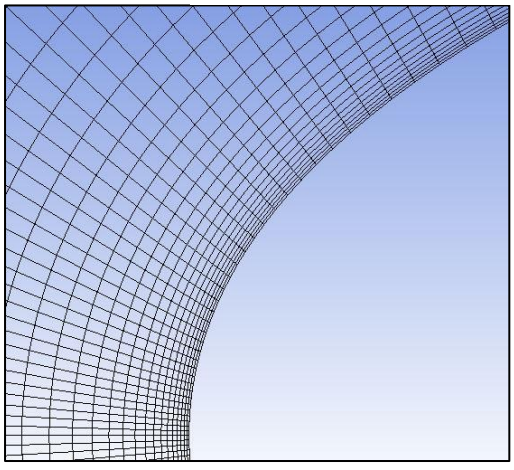


Figure 16: Boundary layer mesh detail

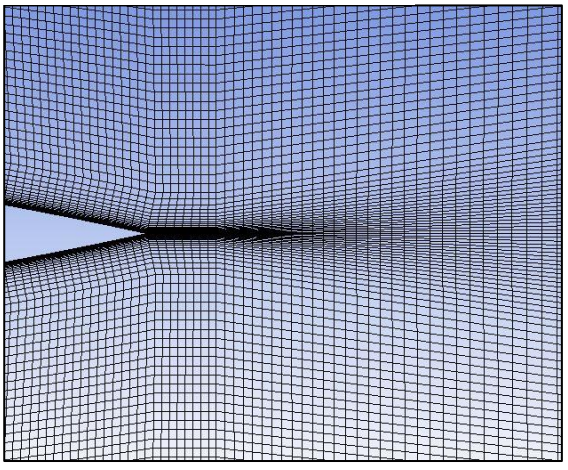


Figure 17: Wake mesh detail

An initial presumption for the thickness of the boundary layer zone was generated using the standard formula for a turbulent boundary layer on a plate.

$$\delta = 0.37x\left(\frac{\mu}{\rho u_0 x}\right)^{1/5} \quad (4.1)$$

Here, δ is thickness (or height) of the boundary layer, ρ is the fluid density u_0 is freestream velocity, x is the distance downstream from the start of the boundary layer and μ is the dynamic viscosity.

The freestream velocity u_0 was set to the maximum operational velocity of 30 m/s. Lower flow speeds would lead to a thicker boundary layer. As a smooth transition between boundary layer and freestream was ensured (with respect to cell height), a “too small” boundary layer zone in case of lower flow speeds would not lead to any problems.

In a second step an initial presumption for the height of the cells closest to the wall, the so called first cell height y_p, must be made. This height is based on the y^+ value.

$$y^+ = \frac{\rho y u_\tau}{\mu} \quad (4.2)$$

$$y_p = \frac{\mu {y^+}_p}{\rho u_\tau} \quad (4.3)$$

${y^+}_p$ is hereby the y^+ value corresponding to the first cell hight y_p. The friction velocity u_τ is hereby defined as

$$\sqrt{\tau_w/\rho} \quad (4.4)$$

where τ_w is the wall shear stress.

The used CFD solver, RANS with k-ω SST model described in detail in section 4.3.2, is able to work in two ways: Either it resolves the boundary layer down to the viscous sublayer, or it uses the well-established wall functions (log law and linear profile) to estimate the flow in the inner layer. Empirically generated data [16] show that the viscous sublayer extends until $y^+ \approx 5$ and that log law region in between $30 \leq y^+ \leq 100$ (see fig. 18). If ${y^+}_p$ is below 5 then, the inner layer is resolved. If it is higher than 30 wall functions are used.

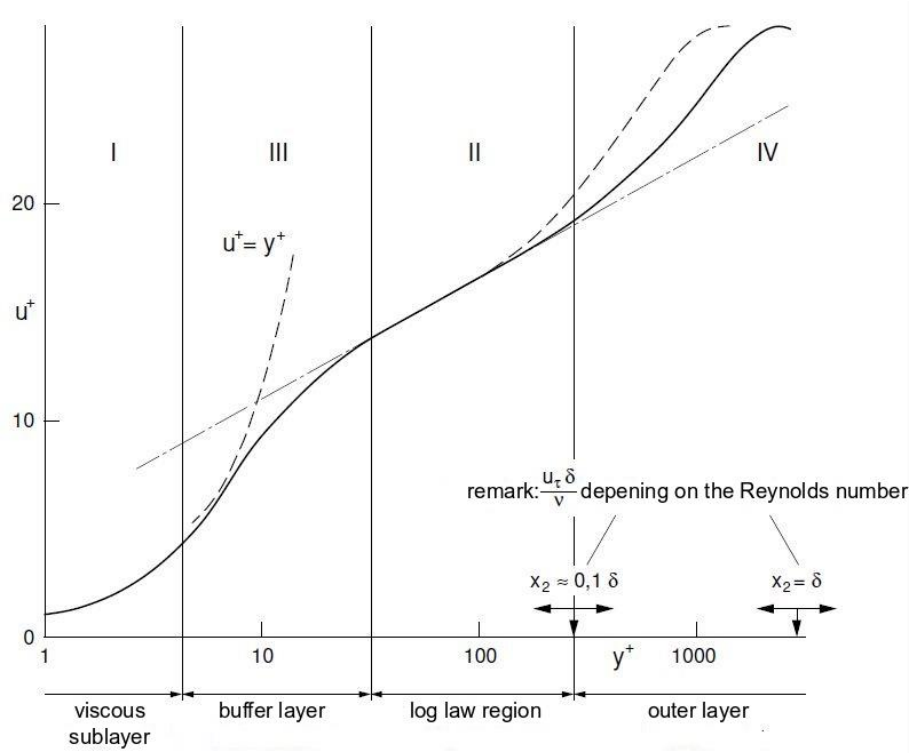


Figure 18: Law of the wall [16]

The first cell height y_p refers in this case to the distance between the wall and the centre of its adjacent cell (see fig. 19) [17]. Note that the Ansys mesher defines its cell height as the overall height of the cell and not as the distance between is centre to its edge. Therefore, the following relation applies:

$$y_h = 2 * y_p \tag{4.5}$$

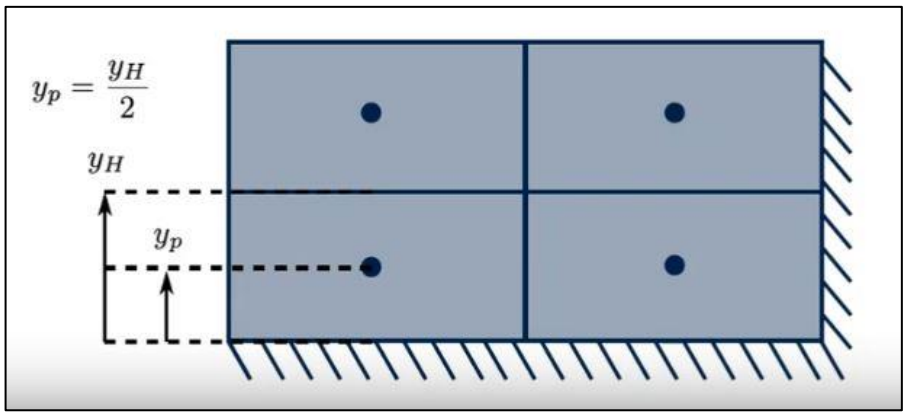


Figure 19: First cell height [17]

A ${y^+}_p$ value between 5 and 30 is to be avoided at all cost, as a first cell height in this region does not allow for a good approximation with wall functions, nor does it allow for a resolved inner layer. Typically, ${y^+}_p \approx 1$ or ${y^+}_p \geq 30$ is targeted. For the present simulation a resolved boundary layer and thus ${y^+}_p \approx 1$ was targeted for the gust vanes and the test wing, whereas wall functions were considered as being accurate enough for the wind tunnel walls leading to a targeted ${y^+}_p \approx 50$ (including some margin). The simple reason being that the boundary layer at the tunnel wall is not very

much of interest and thus computational time could be saved there. The dimensionless y^+ is dependent on flow conditions. To transform a desired y^+ into actual dimension requires some assumptions and multiple calculation steps. It is dependent on flow speed. The initial guess was calculated using the maximum freestream velocity of 30 m/s. To facilitate this calculation, an online calculator with well documented formulas was used [18]. This initial guess must not be perfect, as the first cell height is tuned iteratively as further described in section 4.3.3.

If the simulation was run on lower flow speeds, the height corresponding to $y^+ = 1$ would increase, and subsequently the ${y^+}_p$ would decrease (if the mesh stays the same). This would lead to a better resolved boundary layer if no wall functions are applied (${y^+}_p \leq 5$) but could lead to problems if wall functions should be applied (${y^+}_p \geq 30$). As in the current simulation the latter case is only present at the tunnel walls which are not of special interest, this is acceptable. Additionally, the boundary layer grows in stream-wise direction at any given speed and therefore the ${y^+}_p$ value changes as well, given the height of the first cell is kept constant alongside a wall. The values were tuned to be accurate in the regions of interest: For the wind tunnel wall it was ascertained that the value stays in between $30 \leq {y^+}_p \leq 100$ with a target value as close to 50 as possible over the whole length. For the test wing the ${y^+}_p$ at the trailing edge was targeted to be around 1 whereas higher values were accepted towards the leading edge. The calculated values and initially implemented values can be seen in table 6.

Mesh	**Max. boundary layer thickness δ calculated/implemented**	**Initial first cell height y_h Calculated/implemented[2]**
Wind tunnel (background)	24 mm (29 mm[3])/30 mm	1.2 mm/1.16 mm
Gust vane	2.7 mm/3 mm	0.019 mm/0.017 mm
Test wing	4.7 mm/5 mm	0.02 mm/0.018 mm

Table 6: Boundary layer meshing values

The cell height in the boundary layer zone increases steadily towards the free stream zone. A growth rate of 1.2 [19] was targeted as well as matching cells at the transition to the outer zone.

The overall mesh quality was assessed with three parameters: maximum aspect ratio, maximum skewness and minimal orthogonality. In accordance with the Fluent User's Guide chapter III.6.2.2. [13] and the Meshing 2020 R1 User's Guide chapter “Skewness” [20], these parameter were set as follows:

Maximum aspect ratio: as low as possible, <35 [21]

Maximum skewness (category good): 0.5

Minimal orthogonal quality: 0.01 with significantly higher average of 0.5

4.3.2 Simulation Set-Up – Steady State

- **Solver**
  The steady state simulation was performed as a pressure-based RANS simulation using the $\kappa - \omega$ SST model, which is a model generally recommended for simulations containing airfoils. It

[2] The implemented values differ slightly from the calculated values due to some mathematical restrictions on how the boundary layer can be divided in a natural number of cells

[3] 750mm inlet length. Originally calculated with 450mm inlet length

combines the strengths of the standard $\kappa - \varepsilon$ and the standard $\kappa - \omega$ model. In principle, it uses the $\kappa - \omega$ model to calculate the flow in the boundary layer and the $\kappa - \varepsilon$ model to calculate the free stream flow, as they each produce more accurate solutions in their respective domain. The two models are then blended into each other by blending functions F_1 and F_2 [22], [23]**.** The solving parameters were left on default, as these are empirically generated values.

The k-ω SST model offers multiple additional options which were initially left enabled/disabled as recommended by the default settings:

1) **Low-Re corrections**: Not recommended to be used
2) **Viscous heating**: Not needed for incompressible flows
3) **Curvature correction**: Not needed, as the flow in the present simulation can be considered as not highly curved.
4) **Production Kato-Launder**: Only needed in combination with Intermittency Transition Model
5) **Production Limiter**: Enabled by default
6) **Intermittency Transition Model**: Could increase accuracy, as it helps to model laminar/turbulent transition.

All the information about these options are in accordance with Fluent User's Guide chapter III.12.2.1.3 [13]. The intermittency Transition Model was enabled at a later stage to investigate its influence on the simulation results. A baseline simulation with the option disabled was executed and then repeated with the only change being the enabling of the intermittency transition model and the Kato-Launder production limiter. Two simulations were performed with second and first-order spatial discretisation of the intermittency, but convergence could not be reached with neither of them (see fig. 20) and thus the intermittency transition model and the Kato-Launder production limiter were both disabled again for all subsequent simulations.

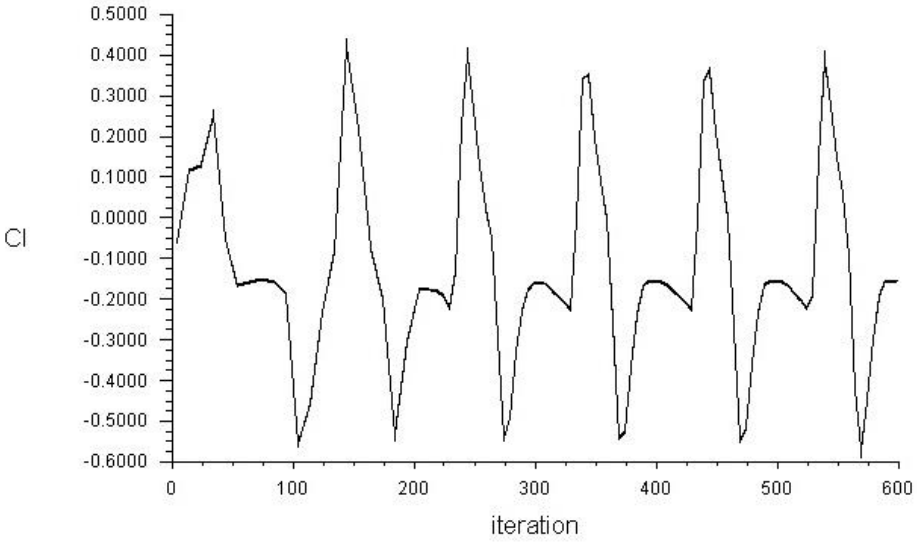


Figure 20: Convergence failure with enabled intermittency transition model

- **Solution Controls**
  The solution controls were left as set by default, as these are based on empirical observations. The solution methods were also mostly left as set by default. Only the spatial discretization was changed to Second Order Upwind as this is supposed to increase accuracy on the cost of higher computational demands as described in the Fluent User's Guide chapter III.73.2.1. [13]. Initial simulation showed that the simulation could still be done in a reasonable amount of time. Changing any other setting would have potentially decreased the simulation accuracy.

  The criteria for a converged solution with regard to residuals were set to be 10^{-2} times smaller than the default value suggested by Fluent, leading to an absolute criteria of 10^{-5} for all residual equations except for the energy residual equation which then is 10^{-8} these criteria are applied as absolute to globally scaled residuals, meaning that that Fluent sums up the imbalance (residual) of all cells of a given quantity, divides this value by the sum of said quantity and compares it to the set convergence criterion. If the criterion is met, the quantity is treated as converged. For further detail check Fluent User's Guide chapter III.48.2.81. and chapter III.37.15.1. [13].

  Additionally, the convergence of lift and drag coefficients for all present airfoils were set as a condition for a converged solution. The solution was set to be considered as converged with regard to a certain parameter if the difference over the last 5 iterations was less than 0.01% of said value. This criterion was set the same for all lift and drag coefficients. For further detail check Fluent User's Guide chapter III.37.16.1. [13].
- **Boundary Conditions**
  An important part of every simulation are correctly set boundary conditions. table 7 summarises the boundary conditions as applied for all simulations performed as part of the present work.

Boundary Condition	**Value**
Inlet (velocity inlet)	
Velocity mag. (uniform distribution at Inlet)	30 m/s
Turbulent intensity	0.005 (0.5%)
Turbulent viscosity ratio	5
Initial gauge pressure	101325 Pa
Outlet (Pressure Outlet)	
Gauge pressure	101325 Pa
Backflow pressure spec.	Static pressure
Backflow turbulent intensity	0.0005 (0.05%)
Backflow turbulent viscosity ratio	1
Walls	
Shear condition	No slip
Wall motion	Stationary wall
Wall sand-grain roughness	
WT	2.9 µm/27.6 µm/0.2 µm (inlet/contr./testsec.)
GV	3.5 µm
TW	5.9 µm

Table 7: Boundary conditions

Non-listed values were kept as default. The turbulence intensity was set according to the data available online for the W-tunnel at TU Delft. The viscosity ratio was set to 5 as typically values between 1 and 10 are used. The same values for the backflow were set to be significantly lower. In the present

system/simulation, backflow can be considered impossible, therefore these values are not of great importance. However, if backflow would occur it would be the non-turbulent air in the surrounding area of the wind tunnel that generates the backflow, hence the low values.

The sand-grain roughness can be derived as follows [24], where R_a is the arithmetic average roughness:

$$K_s \approx 5.863 * R_a \quad (4.6)$$

The sand-grain roughness values were derived from values found in literature [25]–[27] or were estimated.

They only affect the law of the wall, as seen in Fluent User's Guide chapter III.7.4.15.2.8. [13] The non-dimensional roughness height is defined as:

$$K_s{}^+ = \frac{\rho K_s u^*}{\mu} \quad (4.7)$$

u^* is defined as:

$$u^* = C_\mu{}^{1/4} * k_p{}^{1/2} \quad (4.8)$$

y^+ is defined in a similar manner as seen in equation 4.2 The only difference in the definition of $K_s{}^+$ and y^+, apart from the reference value K_s and y, is u_τ being used instead of u^*. In most cases one can assume:

$$u^* \approx u_\tau \quad (4.9)$$

This can be verified by evaluating the CFD simulation for y^+ and y^*, which use u_τ and u^* respectively, and comparing the two. If $u^* \approx u_\tau$ is considered to be true one can conclude that the following is true if similar flow conditions are present:

$$y^+ \widehat{=} K_s{}^+ \xrightarrow{yields} y \widehat{=} K_s \quad (4.10)$$

The Fluent User's Guide chapter III.7.4.15.2.8.[13] further states that the roughness has only an effect on the law of the wall if $K_s{}^+ > 2.25$. With the established correlation between y and K_s in equation 4.9 one can say that as long as equation 4.11 is true, then the wall function is not affected.

$$K_s \leq 2.25 * y_p \quad (4.11)$$

Herby y_p must correspond to a value of y^+ smaller or equal than 1.

The wind tunnel wall is the only wall boundary that must be considered, as it is the only one where wall functions are applied. The y_h listed for the wind tunnel in table 6 corresponds to a y^+ value of 50. Therefore, it needs to be divided by 50 and again by 2 according to equation 4.5. This leads to a value of 12 µm. Multiplying this by 2.25 according to equation 4.11 leads to a value of 27 µm. The biggest K_s value of the wind tunnel wall is 27.6 µm and thus will lead to a slightly distorted wall function. The other values are substantially smaller. In conclusion one can say that these roughness values either have no influence on the simulation at all (gust vanes and test wing), are small enough to not change the wall function (inlet and test section) or only influence the wall function slightly (contraction). Therefore, it was considered to be unnecessary to have more exact values, than the ones used or to further investigate the topic.

4.3.3 Mesh Convergence

A mesh convergence study is performed to ensure, that a simulation leads to results independent of the mesh. In the present case, this was performed using steady state simulations. An initial mesh is generated and a simulation is executed to set a baseline. Subsequently, the mesh is refined and thus the cell count increased. The simulation is then repeated with otherwise the same set-up. Physical values of interest are logged and compared between different iterations. This process of mesh refinement and simulation is repeated until the logged values converge. At this point further mesh refinement does not influence the results of the simulation anymore and mesh convergence is reached. The mesh used in the second last simulation is usually the one used to save computational cost, as the last iteration does not deliver any significantly more accurate results. In the present case the lift (Cl) and drag coefficient (Cd) of the upper gust vane were used as reference values. While performing the mesh convergence study, incorrect physical reference values were used (see fig. 21). The reference values are used to calculate physical values during the postprocessing of the simulation results. Hence, they do not affect the simulation results themselves (including the y^+ values), but the post processed data such as Cl and Cd. The systematic error in the Cl and Cd calculation is present in all iterations of the mesh convergence study as the incorrect reference values were not changed. The performed simulation could therefore still be used for the mesh convergence study, as it only aims to ensure results independent of the mesh. This systematic error was resolved in a later step by replacing the incorrect reference values with the correct ones (see fig. 21). The virtual depth of the simulation was changed from the 0.4 m of the actual system to unit length (1 m) to get a correct 2D Cl and Cd value. The chord length was changed to the actual chord length of 80 mm.

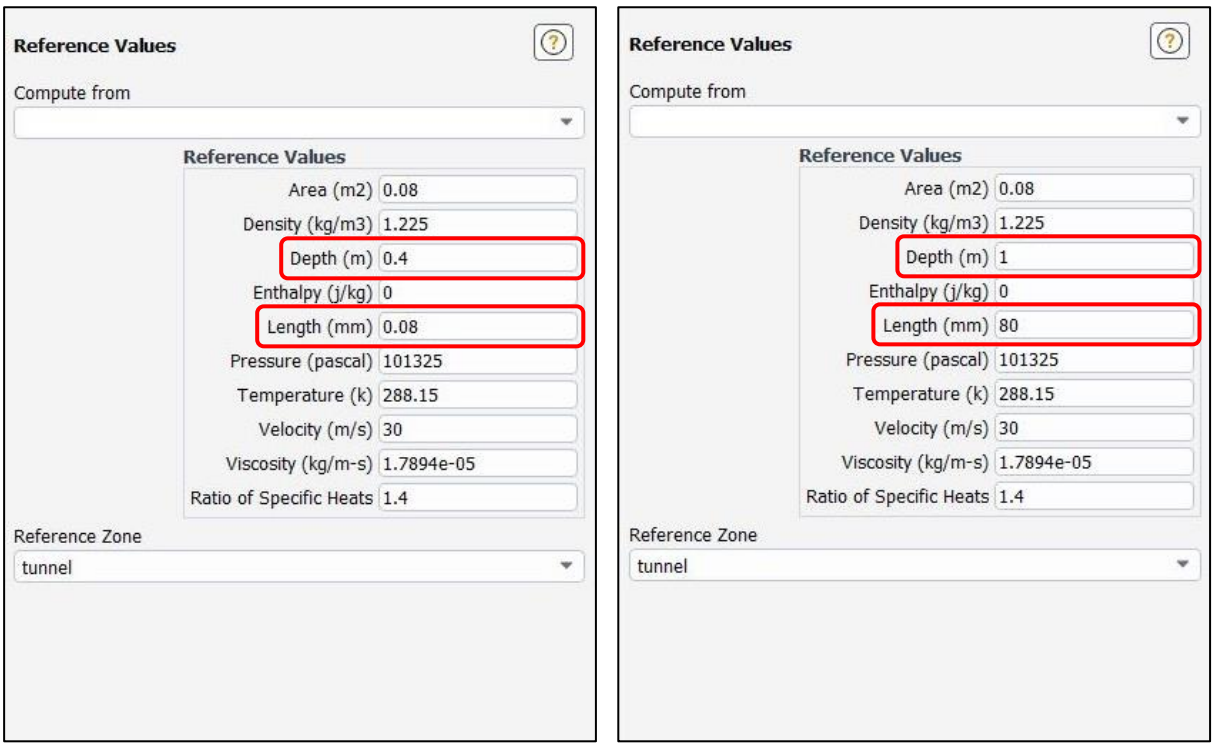


Figure 21: Incorrect vs correct reference values

While performing the mesh convergence study, the cell heights in the boundary layer were adapted simultaneously to reach the desired y^+_p values.

The mesh convergence study was performed at an angle of attack of 15°. The numerical results of the mesh convergence study can be seen in table 8:

Run	1	2	3	4	5
GV Angle [deg]	15	15	15	15	15
Cell count	47160	95260	162420	306800	237780
WT	8960	18200	40200	119800	89056
GV	10640	19700	31520	54340	42872
TW	16920	37660	59180	78320	62980
$\boldsymbol{y^+_p}$					
WT	33.40-55.33	34.40-58.85	33.63-57.52	33.53-57.79	33.52-57.76
GV	0.142-4.126	0.092-3.843	0.117-4.777	0.123-4.761	0.123-4.762
TW	0.588-2.359	0.579-2.614	0.58-2.613	0.555-2.598	0.421-1.958
Cl/Cd[4]	0.4344/0.0331 9	0.4712/0.031 3	0.4478/0.0339 6	0.4518/0.0333 1	0.4525/0.0333 3
Iterations	790	473	316	2500	1245
Sim.Time [min]	n/a (<10)	n/a (<5)	18	68	27
Mass imbalance					
m_{totin} *[kg/s]*	5.88	5.88	5.88	5.88	5.88
Δm *[kg/s]*	0.00345	0.00383	0.0028	0.0035	0.0024
Δm *[%]*	0.059	0.065	0.048	0.059	0.041

Table 8: Mesh convergence

The changes made between each run can be found below.

- Run 1: Initial run
- Run 2: The cell count in all meshes was increased in both directions to nearly double the total cell count
- Run 3: y^+_p was increased at the GV and the transition between boundary layer and freestream was smoothened. The cell count was nearly doubled by decreasing cell size in x - direction for the WT mesh and in y - direction for the GV and TW meshes.
- Run 4: The cell count was roughly doubled by decreasing cell size in x - direction for the WT mesh and in y - direction for the GV meshes. The wake of the GV meshes was additionally refined in x – direction. The TW mesh was refined in both directions. The GV and TW meshes were overall improved with regard to size matching between different mesh zones. Mesh convergence was achieved at this point.
- Run 5: Final mesh. The cell count was again reduced to a value between run 4 and 3, additionally the mesh was once again overall smoothened to improve different various mesh quality parameters mentioned in section 4.3.1.

It was defined that the simulation is considered to be independent if the Cl value of the upper gust vane differs less than 1% when the cell count is doubled between two consecutive simulations. The same condition was also applied to the Cd value of the upper gust vane, but 2% difference were accepted as it is considerably more difficult to get an accurate result for the Cd value.

[4] These values were logged for GVU

In mathematical terms these conditions can be formulated as follows:

$$\frac{2 * |Cl_i - Cl_{i-1}|}{Cl_i + Cl_{i-1}} \leq 0.01 \qquad (4.12)$$

$$\frac{2 * |Cd_i - Cd_{i-1}|}{Cd_i + Cd_{i-1}} \leq 0.02 \qquad (4.13)$$

As mentioned above, these conditions were met with run 4, after doubling the cell count for the third time after the initial run, with $\Delta Cl/Cl$ = 0.9% and $\Delta Cd/Cd$ = 1.9%, as shown in figure 22 to 25.

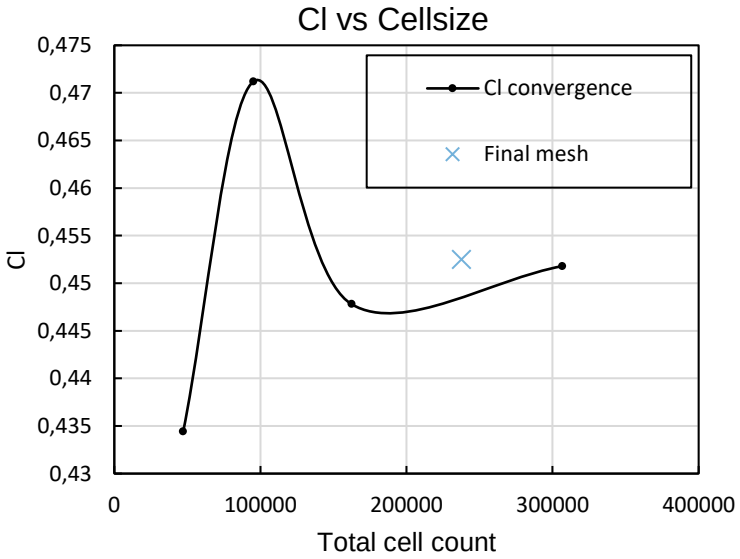


Figure 22: Cl convergence

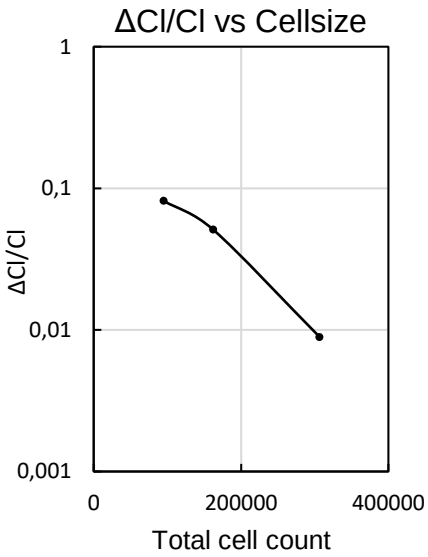


Figure 23: ΔCl vs. cell count

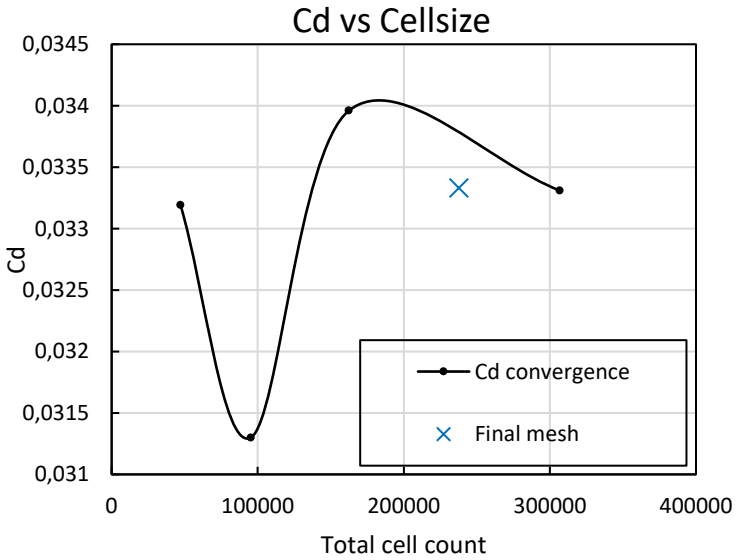


Figure 24: Cd convergence

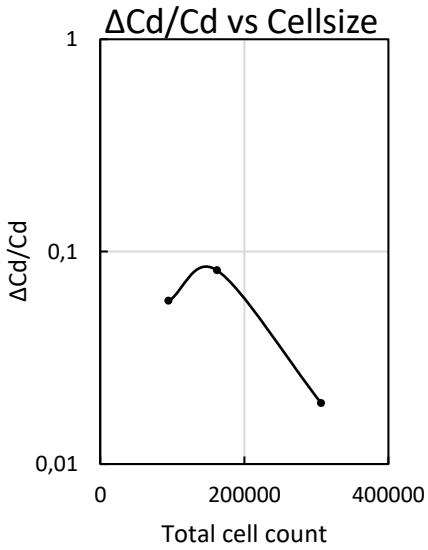


Figure 25: ΔCd vs cell count

The mesh convergence study did not require many steps, as a structured and reasonably refined mesh was generated already for the first run.

The mesh quality parameters mentioned in section 4.3.1 were all met by the final mesh, with the exception of the aspect ratio in the case of the test wing mesh, which went up to 56,3. However, this is not problematic, as quad cells can feature higher aspect ratio, as long as they are aligned with the flow as mentioned in Fluent User's Guide chapter III.6.1.3.2 [13]. This is the case, as these high aspect ratios are only present in cells at the proximity of the test wing surface (inner boundary layer) (see fig. 26).

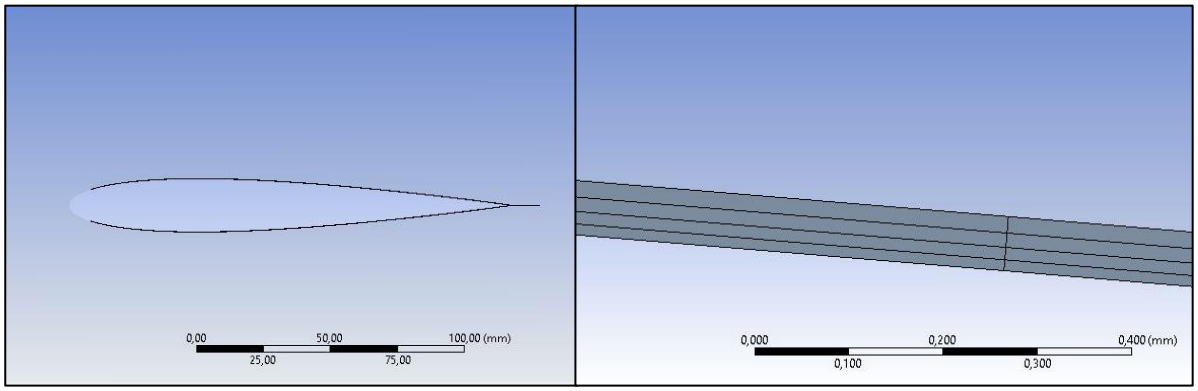


Figure 26: High aspect ratio cells in boundary layer

Finally, the simulation was updated in two ways after the mesh convergence study was completed:

- The inlet length was changed from 0.4 m to the 0.75 m as mentioned at the beginning of this section. This led to a new and final cell count of the background mesh of 110176 leading to a total of 258900. This did neither affect the size nor the shape of the cells used in the background mesh, as only cells were added to elongate the inlet section.
- The new and correct reference values were applied as mentioned in the beginning of this section.

The above-mentioned changes were kept for all subsequent simulations

4.3.4 Flow Characteristics at Different Angles of Attack

It became evident, that the freestream flow deflected towards the centre by the constricting shape of the gust generator enclosing, affects the flow around the gust vanes significantly (see fig. 27).

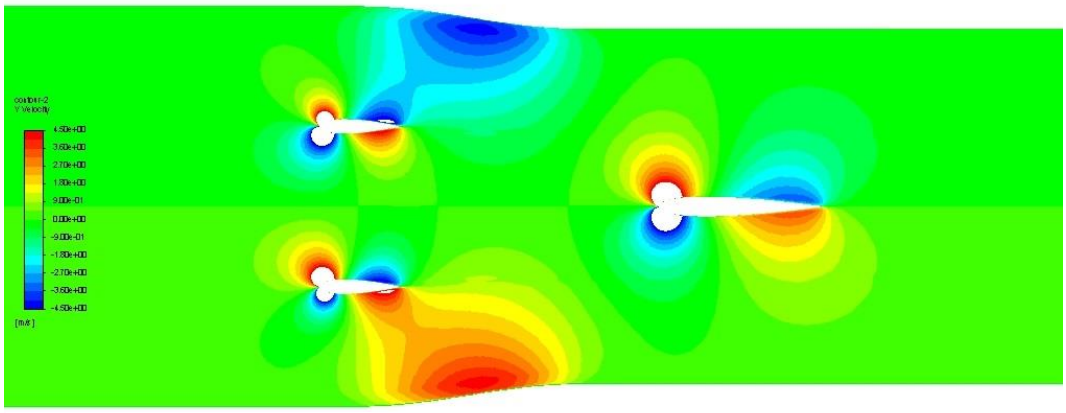


Figure 27: v_y at 0° angle of attack

To further investigate the flow characteristics, a steady state simulation was performed with gust vane angles in 2.5° steps between -15° and +15°. The numerical results of these simulations are summarized in the table below.

GV Angle [deg]	**Cl** *GVU* *GVL* *TW*	**Cd** *GVU* *GVL* *TW*	y^+_p *WT* *GV* *TW*	**Mass imbalance** m_{totin} *[kg/s]* Δm *[kg/s]* Δm *[%]*	**Con-verged**	**Simulation Time [min]**
-15	-0.9848 -1.1785 0.0851	0.08723 0.0869 0.04023	30.93-57.75 0.106-4.809 0.422-1.974	14.7 0.0066 0.045	No[5]	60
-12.5	-1.0353 -1.067 0.0775	0.0443 0.06174 0.04012	29.74-57.75 0.09-4.478 0.42-1.964	14.7 0.0074 0.05	Yes	9
-10	-0.9535 -0.8628 0.0656	0.02716 0.0478 0.0402	30.11-57.75 0.073-4.117 0.418-1.952	14.7 0.0075 0.051	Yes	7
-7.5	-0.798 -0.6119 0.0504	0.02114 0.03873 0.04036	31.06-57.75 0.077-3.82 0.418-1.938	14.7 0.0063 0.043	Yes	6
-5	-0.6013 -0.3531 0.034	0.01975 0.03201 0.04054	32.23-57.75 0.078-3.483 0.418-1.923	14.7 0.0003 0.002	Yes	6
-2.5	-0.3809 -0.0997 0.0171	0.02055 0.02682 0.04063	33.48-57.75 0.108-3.13 0.418-1.907	14.7 0.0002 0.001	Yes	7
0	-0.1457 0.1457 0	0.02301 0.02301 0.04066	34.69-57.75 0.118-2.776 0.418-1.893	14.7 0.0003 0.002	Yes	8
2.5	0.0997 0.3809 -0.0171	0.02682 0.02055 0.04063	33.5-57.75 0.108-3.13 0.418-1.908	14.7 0.0001 0.001	Yes	6
5	0.3531 0.6013 -0.034	0.03201 0.01974 0.04054	32.25-57.75 0.079-3.483 0.418-1.923	14.7 0.0013 0.009	Yes	6
7.5	0.6189 0.82 -0.0509	0.03899 0.01857 0.04034	30.62-57.75 0.085-3.844 0.418-1.938	14.7 0.0039 0.027	Yes	6
10	0.8629 0.9535 -0.0656	0.0478 0.02716 0.0402	30.12-57.75 0.074-4.117 0.418-1.952	14.7 0.0068 0.046	Yes	7
12.5	1.067 1.0353 -0.0775	0.06174 0.0443 0.04012	29.74-57.75 0.09-4.478 0.42-1.964	14.7 0.0066 0.045	Yes	8
15	1.1792 1.0015 -0.0854	0.08689 0.08712 0.04022	30.46-57.75 0.096-4.819 0.422-1.975	14.7 0.006 0.041	No[6]	59

Table 9: Results of steady state flow simulation over angle of attack range

[5] Periodically stable, more iterations would not have led to convergence.
[6] Periodically decreasing, more iterations would have led to convergence.

A baseline for the lift was generated using XFoil with a Reynolds number of 164300 and a N_{crit} value of 9, which can be expected for an average wind tunnel [28]. The XFoil data can be found in appendix A. Figure 28 and 29 show the results of this study.

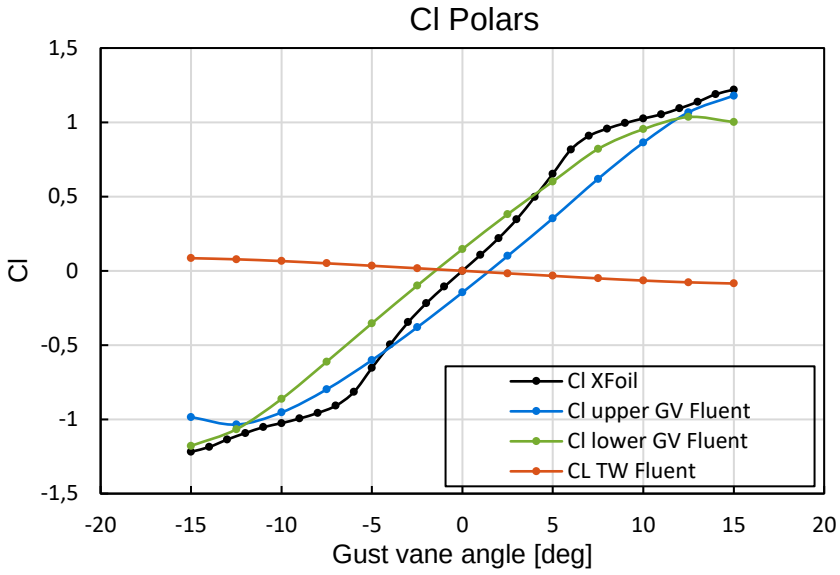


Figure 28: Steady state Cl polars

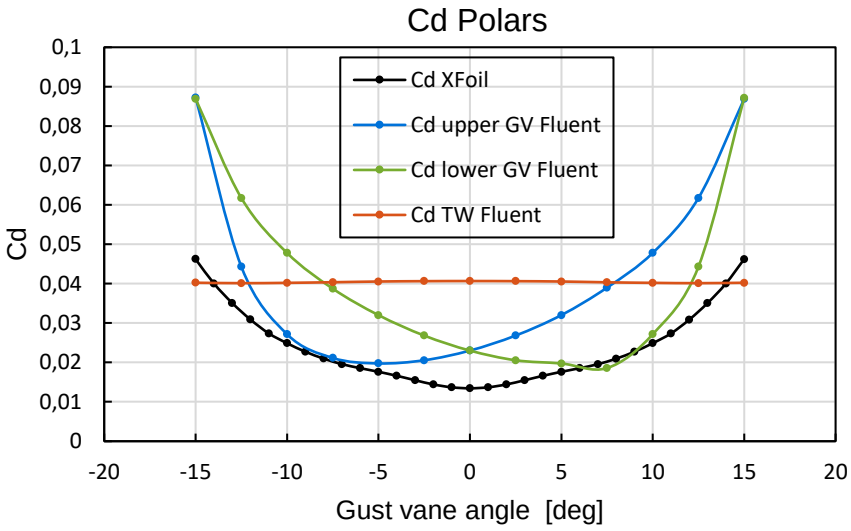


Figure 29: Steady state Cd polars

Both charts indicate multiple phenomena:

- The Cl curves generated with data from Fluent have a slope similar to the one generated by XFoil, indicating that the data generated with Fluent is reliable.
- The lift of the test wing decreases (correctly) as the lift of the gust vanes increases, as it is in the wake of the gust vanes.
- As mentioned earlier and well visible in figure 28, both gust vanes are exposed to a flow with a vertical component. As this vertical component is pointing towards the centre of the contraction, the upper gust vane experiences a negative angle of attack at a geometrical angle of 0°, whereas the lower gust vane experiences a positive angle of attack under the same

circumstances. These flow conditions are reflected in the lift curves as they are shifted towards a positive gust vane angle for the upper gust vane and a negative gust vane angle for the lower gust vane, as they generate zero lift when they are aligned with the surrounding flow.

- The Cd curves are shifted as well, but counterintuitively not in the same direction as the Cl curves. The reason for this behaviour is based again on the curved flow: The lift force of a wing is perpendicular to the flow direction around it. If the upper gust vane at a geometric negative angle is taken as a reference, the lift is pointing downwards and slightly forward (perpendicular to the flow). The part of the lift force pointing forward counteracts some of the drag force calculated only in positive x - direction, leading to the lowest drag force at a slightly negative angle of attack. The same principle applies to the lower gust vane, just with inversed signs.

A simple sanity check of the results of the simulation was done. The absolute lift and drag coefficient at a certain positive angle of attack for the upper gust vane should be the same as the absolute lift and drag coefficient of the lower gust vane at the corresponding negative angle of attack. This is due to the symmetry of the problem at hand about the x -axis. To assess this, simply the difference between each pair of values was calculated as seen in table 10.

GV angle (GVU) [deg]	**Cl Symmetry** ($Cl\_GVU_n + Cl\_GVL_{-n}$)	**Cd Symmetry** ($Cd\_GVU_n - Cd\_GVL_{-n}$)
-15	0.0167	0.00011
-12.5	0	0
-10	0	0
-7.5	0.022	0.00257
-5	0	1E-05
-2.5	0	0
0	0	0
2.5	0	0
5	0	0
7.5	0.007	0.00026
10	1E-04	0
12.5	0	0
15	0.0007	-1E-05

Table 10: Symmetry assessment of Fluent results

If the value is zero or close to zero, the absolute values are considered to be equal and therefore pass the check. For most of the values this is true. It can however be seen, that this is not given for two simulations. At -7.5° and at ±15° some discrepancy can be detected. The discrepancy at -7.5° corresponds to the dent in the graph as seen figure 29. The reason for this dent could not be evaluated. The discrepancy at ±15° is most likely the consequence of the non-converged simulations at this angle of attack, especially at -15° as this simulation was periodically stable.

In summary, it can be said that the shape of the contraction, the proximity of the walls as well as the interaction between the two gust vanes lead to a changed shape of the lift as well as of the drag curves. Therefore, these curves cannot directly be compared quantitatively to the ones from XFoil.

4.4 Transient Simulations

After the steady state simulations were considered to be well tuned, the simulation set-up was changed to a transient simulation for all the subsequent simulations.

4.4.1 Simulation Set-Up – Transient

To switch from steady state to transient simulations some adaptations to the simulation set up were necessary, the most obvious being the actual switch from a steady state solving routine to a transient solving routine. Nevertheless, a major part of the steady state set-up was kept unchanged for the transient simulation. The adaptations made are described below:

- **Mesh motion**
  Use of the overset approach to generate the mesh, as described in section 4.2, allowed for a straightforward implementation of the mesh motion. Each gust vane mesh was moved using user defined functions. These functions take three inputs: the displacement in y - and x - direction from an initial position and the frequency of the motion. The maximum amplitude was fixed at 10°. The displacement parameters were necessary to allow for a parametrised simulation setup for the optimisation described in section 5. The Motion described by the UDF would first displace the gust vane mesh to its desired position, then wait a 3/5 of a period (equal to 0.05 second at the highest frequency of 12Hz) to ensure steady initial conditions. At last the periodic motion starts for as long as the simulation would run. In case of a 1-cos gust the UDF is slightly different in the way that it does not allow for a gust vane displacement (as the optimization was done only with sin gusts) and only one movement is executed. The UDF code can be found in appendix C.
- **Solver**
  Apart from switching the solver from steady to transient, no changes were made compared to the set-up used for the steady state simulations.
- **Solution controls**
  The solution controls were left unchanged as well. The convergence criterion was changed to be applied for each time step (time step convergence). A time step was considered converged if the difference between two consecutive iterations for the Cd and Cl values of all involved airfoils was less than 0.01% of said value. Residual convergence was relaxed with respect to the steady state solution to the default values of 10^{-3} resp. 10^{-6} as otherwise convergence could hardly be reached. The convergence was then dominated by the Cl and Cd values. The time step size was chosen to be constant, for more details see section 4.4.2.
- **Boundary conditions**
  The boundary conditions were left unchanged.

4.4.2 Time Step Size Convergence

Similar to the mesh convergence study performed at steady state, a convergence study related to the time step size is necessary in the case of transient simulations. The time step size determines how many steps are used to simulate a time dependent flow of a fixed duration. The sample rate convergence study is performed to ensure that the simulation result is independent of the number of time steps which are used for a given simulation. A baseline is set with a simulation using an initial time step size. Subsequently, the time step size is decreased with each iteration and thus the number of time steps is increased. Otherwise no changes are made to the simulation set-up. Physical values of interest are logged and compared between two consecutive iterations. This procedure is repeated until the difference of the logged values between two consecutive simulations converges. The time step size can then be chosen equal to or smaller as the second to last simulation. It must be noted that the time step size is linked to the convergence behaviour of the time step itself. A smaller time step size converges generally faster than a larger one [29]. As the total number of iterations performed per simulation is the sum of the iterations performed on each time step, the computational effort can be smaller even if a

smaller time step size is chosen, as the gain of lesser iterations per time step can outweigh the increased number of time steps. It can be concluded that the time step size must be smaller than the one of the second to last step in the time step size convergence study to ensure a result independent of the time step size, but the second to last time step size does not necessarily deliver the least computational effort.

In the present case, the convergence was assessed with the Cl value of the upper gust vane over time (see fig. 30) as well as with the gust velocity at maximum gust vane deflection at x = 70 mm (see fig. 31). This way the convergence is assessed in a temporal as well as in a spatial manner. 50 time steps for one motion period was set as a baseline. The results of this study can be seen in table 11.

Run	1	2	3	4
Time Steps	50	100	150	200
Freq [Hz]	12	12	12	12
Max GV angle [deg]	10	10	10	10
$\Delta v_y max$ average [%]	-	5.63	2.89	1.55
ΔCl_{GVU} average [%]	-	13.06	4.8	2.43
Sim. Time [min]	28	31	44	51

Table 11: Time step study

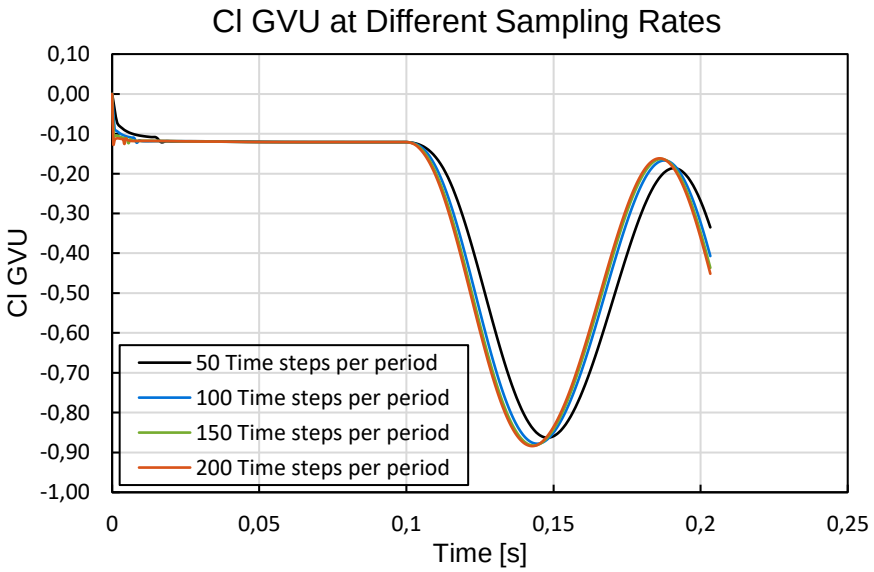


Figure 30: Coefficient of lift over time for multiple sample rates

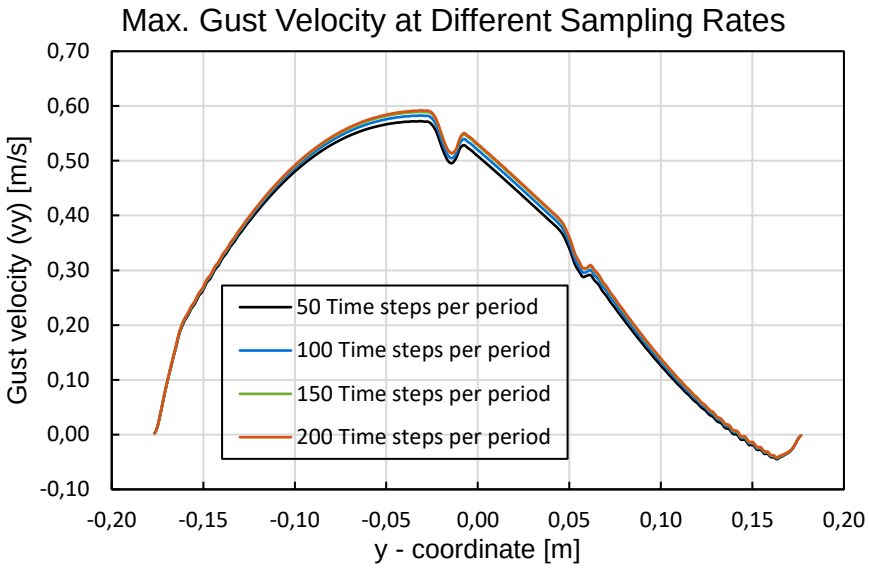


Figure 31: Gust velocity distribution over y for multiple sample rates

It was defined that the simulation is considered to be independent if both values, v_y as well as the Cl value of the upper gust vane differs less than 3% if the number of time steps is increased by 50 per period. In the case of v_y, the resulting values were averaged over y and in the case of Cl over time. Here only values after the start of the motion were considered, as the settling part seen in figure 30 would falsify the result. In mathematical terms these conditions can be formulated as follows:

$$\left(\sum_{j=0}^{n} \frac{2 * |Clj_i - Clj_{i-1}|}{Clj_i + Clj_{i-1}}\right) \Big/ n \leq 0.03 \qquad (4.14)$$

$$\left(\sum_{j=0}^{n} \frac{2 * |v_y j_i - v_y j_{i-1}|}{v_y j_i + v_y j_{i-1}}\right) \Big/ n \leq 0.03 \qquad (4.15)$$

Herby j is the variable corresponding to the individual data points in a single simulation. In the case of equation 4.14 this variable is related to time whereas it is related to the y coordinate in equation 4.15. i on the other hand, corresponds to the different simulations.

Figure 32 and 33 show that the convergence criterion was met at 200 time steps per period. As the simulation was computed substantially faster with 150 time steps, subsequent simulations were performed with at least 150 time steps per period. For motions at high frequencies, this value increased up to 180 to speed up time step convergence.

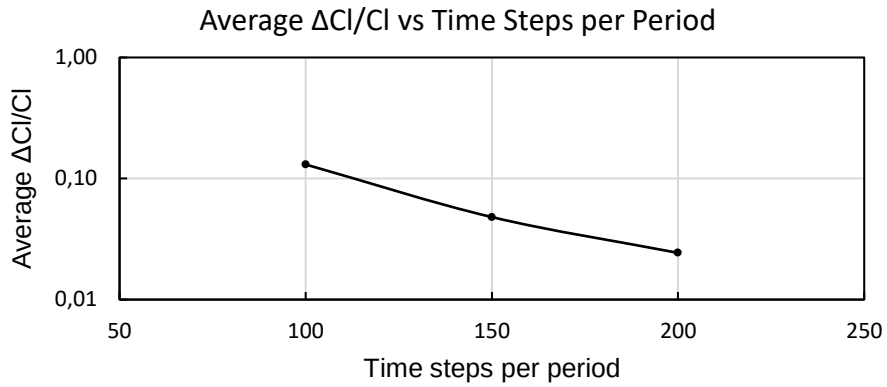


Figure 32: ΔCl vs time steps per period

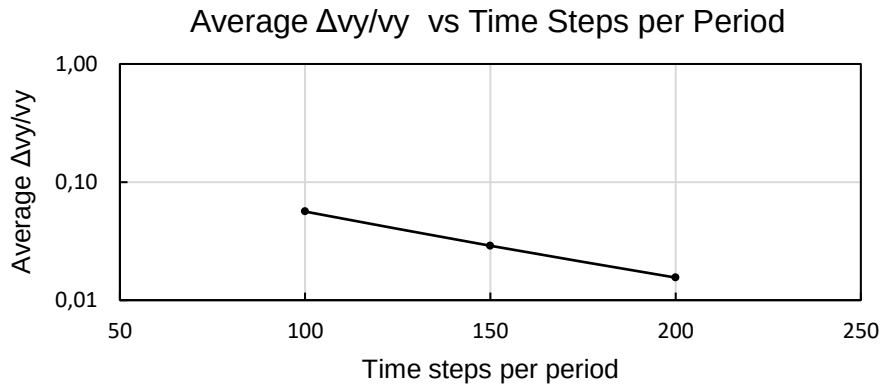


Figure 33: Δvy vs time steps per period

5 Optimisation

As described at the beginning of section 4, an optimisation process was performed to define the optimal position of the gust vanes. Additionally, the influence of the hinge point of the gust vanes on the resulting gust was evaluated based on a concrete flow phenomenon observed on the gust generator built for the OJF.

Two parameters were defined to assess the quality of the gust. The optimisation was performed with regard to these two parameters:

1) **Gust velocity/Gust angle:** The measured velocity in y direction at the point of interest. A large gust velocity is here by desirable, as this will decrease the limitations for later experiments in which the gust generator will be used. The gust velocity in y - direction dominates the gust angle, as v_y is changing substantially more in relative terms as v_x. Subsequently, both measures are applicable as a measure of gust strength.
2) **Gust uniformity:** The measured mean deviation of the gust angle or gust velocity in y-drection at a given time. A small deviation, related to a uniform gust, is desirable. If an airfoil or any other object subjected to the gust during an experiment is able to move in y - drection, it is desirable that the gust experienced is independent of the position of the airfoil.

These two parameters v_{ymax} and $d_{\overline{vymax}}$ were normalized and combined to one optimization parameter opt using a weighted function with the weight w_1 and w_2:

$$opt = v_{ymaxnorm} * w_1 + d_{\overline{vymaxnorm}} * w_2 \quad (5.1)$$

The mean deviation $d_{\overline{vymax}}$ is hereby calculated as follows:

$$\overline{v_{ymax}} = \frac{1}{n}\sum_{i=1}^{n} v_{ymaxi} \quad (5.2)$$

$$d_{\overline{vymax}} = \frac{1}{n}\sum_{i=1}^{n} \left| v_{ymaxi} - \overline{v_{ymax}} \right| \quad (5.3)$$

Fluid simulations were performed with the gust vanes at different positions inside a design space to generate the flow data subsequently used to generate the two parameter mentioned above. All simulations were done in 2D. The following sections describe the optimisation process in detail.

5.1 Hinge Point Position

During the development of the large gust generator for the OJF at TU Delft a counter intuitive flow behaviour was observed. Under certain conditions, the gust velocity pointed in the opposite direction as one would expect. This behaviour was observed at the beginning as well as at the end of a motion [5] forming two dips in the time dependent flow. An initial theory behind this phenomenon related this dip to the motion of the gust vane. With a hinge point behind its leading edge, for instance at 0.25 chord length (c), the part of the gust vane in front of the hinge point will move in the opposite direction than the rest of the vane.

This motion contributes to the angle of attack as seen by the gust vane. It was considered that under certain condition this dynamic contribution could lead to a temporarily negative angle of attack, when the gust vane itself increases its angle as well as vice versa if it decreases its angle of attack (see fig. 34).

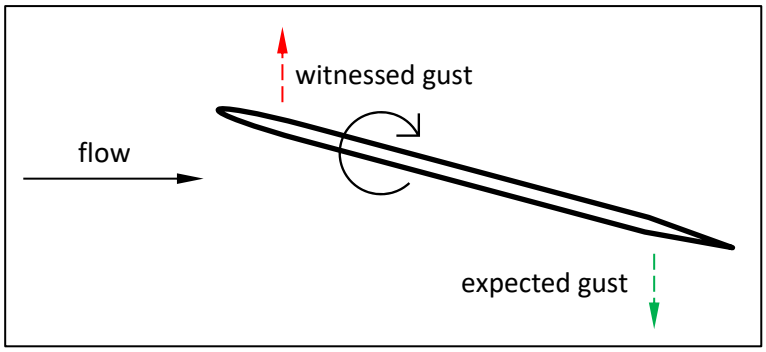


Figure 34: Hinge point related motion

To validate this theory, three simulations of 1-cos gusts were run with different hinge points for the gust vanes, but otherwise the same conditions. A 1-Cos gust at 12Hz with a maximum gust vane deflection of 10° was used in all three simulations. The hinge points were set at 0c 0.25c and 0.5c. The results of these simulations are displayed in figure 35.

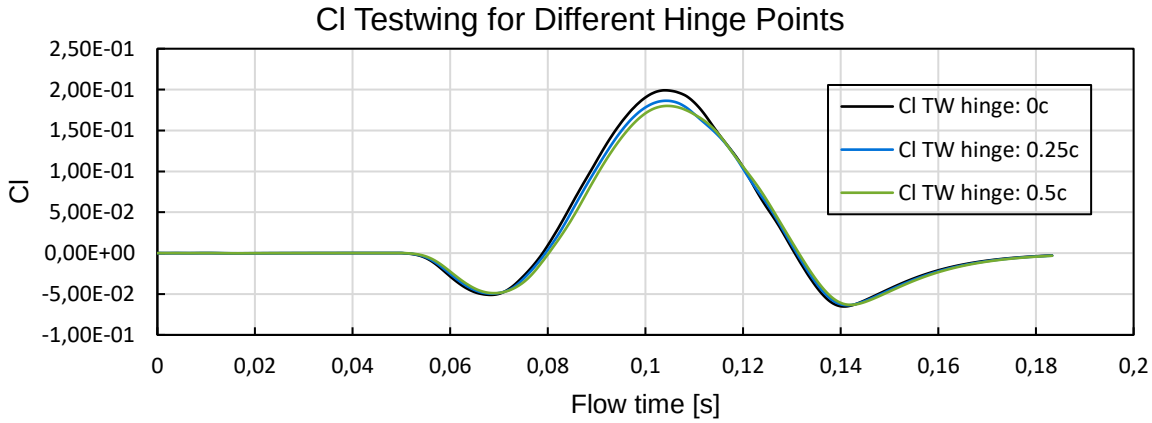


Figure 35:Hinge point related gust inversion behaviour

As it can be clearly seen, the gust inversion is not, or only in a very limited manner, affected by the position of the hinge point, thus the initial theory was proven wrong. Literature moreover suggests that this phenomenon is based on vortices shedding at the beginning and at the end of the motion, if flow is in the unsteady regime. Similar behaviour is seen as the response to a rotational step motion of an airfoil [30]. Later simulations as well as final test results see section 9 proved to be in accordance with this explanation as they demonstrated a strong correlation between the reduced frequency of the flow and the prominence of the inverted gust.

One can further see in figure 35 that a hinge point closer to the leading edge increases the maximum gust angle. A hinge point close to the leading edge also reduces the torque that needs to be generated by the motor, as the flow assists the gust vane motion. For more details refer to section 6. It was subsequently decided that the gust vanes will be hinged at their leading edge.

5.2 Gust Vane Position

After the transient simulation was set up and tuned as well, design optimisation with regard to the placement of the airfoils could be started. As described at the beginning of section 5, the goal of the optimisation was to maximise the gust angel produced by the gust generator as well as to minimise the deviation of the gust velocity and angle in the area of interest in y - direction.

The optimisation was performed in four stages as described below:

1) An initial limited set of simulations was performed to assess the coupling of the actuation frequency with the gust vane position. The goal of this step was to assess if the frequency-related behaviour is similar for different gust vane positions.
2) Additional simulations across the design space are executed to generate a grid of data points.
3) The flow data generated in step two is post processed and interpolated to generate a finer grind of points at each of which the final optimisation parameter is calculated.
4) If the optimal design point is based on interpolated values, a last fluid simulation is performed at said design point to validate the interpolated data.

All simulations were performed at the same flow speed as it is stated in literature that the gust angle is solely dependent on reduced frequency but not on the flow speed itself [4].

5.2.1 Influence of the Gust Vane Position on the Frequency related flow behaviour

As mentioned in section 3.1 the gust generator should be designed to work for entry flow speeds of up to 30 m/s and should cover reduced frequencies up to 0.2, as everything above that is considered as highly unsteady. Three frequencies were defined representing the three different flow regimes as seen in table 12. The frequencies were calculated with a flow speed of 30m/s and the airfoil semi-chord of the test wing of 80mm.

Frequency [Hz]	Reduced Frequency	Flow regime
12	0.2	highly unsteady
4	0.067	unsteady
0.5	0.008	quasi steady

Table 12: Simulated frequencies

A grid was generated to cover most of the range of possible positions. The range was restricted by the requirement that the whole gust vane is at all time inside the gust generator and a deflection of 15° with at least 5° safety margin is possible without colliding with the walls. The grid derived with respect to these restrictions can be seen in figure 36 and all positions are listed in table 13. Note that the gust vanes are always placed symmetrically around y = 0.

x – position [mm]	y – position [mm]	Space between gust vanes [mm]
-84	±40	80
-84	±72	144
-84	±104	208
-84	±136	272
-156	±40	80
-156	±72	144
-156	±104	208
-156	±136	272
-228	±40	80
-228	±72	144
-228	±104	208
-228	±136	272
-300	±40	80
-300	±72	144
-300	±104	208
-300	±136	272

Table 13: Simulated gust vane positions

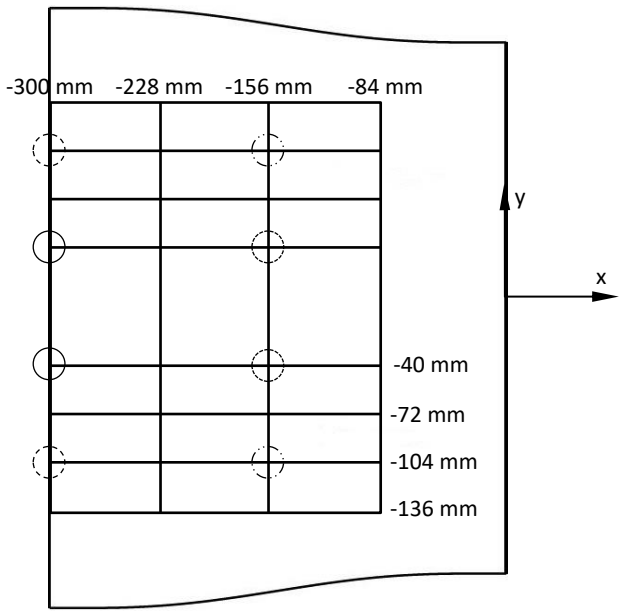


Figure 36: Design Points

To investigate if the behaviour of the flow with regard to the motion can be treated independently of the position of the gust vanes or if the two are linked and influence one another, simulations were run for all motion frequencies mentioned in table 12 and with four different position configurations as marked in figure 36. A sin gust was simulated. The results of these simulations are illustrated in figure 37.

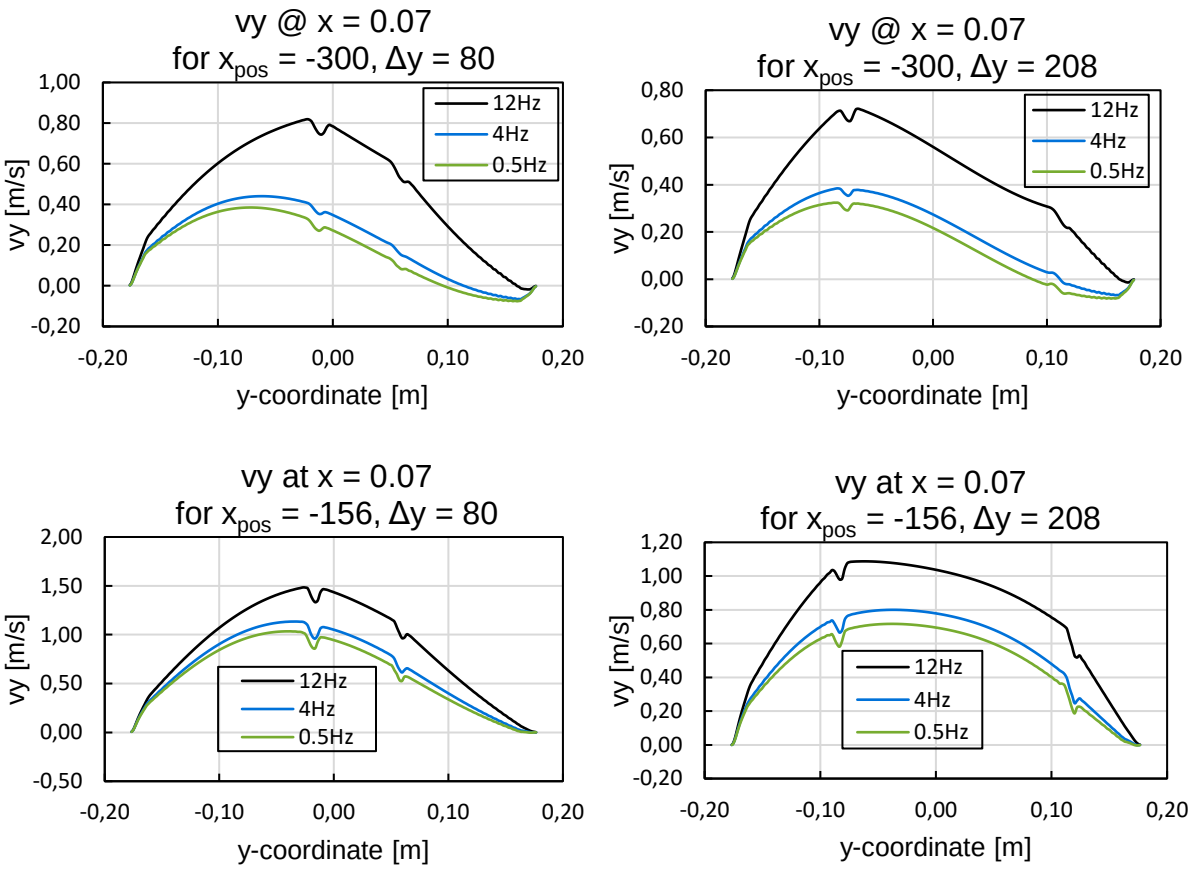


Figure 37: vy at maximum gust vane deflection for multiple design points

It was observed that the general shape of the gust is independent of the motion frequency but strongly influenced by the position of the gust vane. This was expected, as the relative position of the gust vanes to the narrowing walls highly affects the flow which the gust vanes experience. To further assess the coupling of the gust vane position and the motion frequency, the gust velocity at the maximum gust vane deflection was collected for all design points (see tab 14). The gust velocity is hereby the average of the absolute value at the maximum negative and at the maximum positive deflection.

x – position [mm]	Space between gust vanes [mm]	Frequency	Maximum gust velocity [m/s]
-300	80	12	0.792
-300	80	4	0.343
-300	80	0.5	0.274
-300	208	12	0.567
-300	208	4	0.267
-300	208	0.5	0.219
-156	80	12	1.459
-156	80	4	1.027
-156	80	0.5	0.953
-156	208	12	1.057
-156	208	4	0.761
-156	208	0.5	0.702

Table 14: Max. gust velocity for different frequencies and gust vane positions

If the gust velocities at x = -156 mm are normalised with their counterpart at – 300 mm one can reason three things (see fig. 38):

- According to literature [4] the gust velocities drop drastically in down-stream direction meaning the gust decays as it travels away from the gust vanes. Therefore, a gust vane placement closer to the test section will increase the gust strength drastically.
- The gusts decay more or less equally strong for different gust vane spacing. Thus, it can be assumed that changes in the gust related to changes in spacing of the gust vanes are similar for different frequencies and these two parameters can be treated as independent.
- The gust decays faster for lower frequencies. This is mentioned as well in literature [4]. As a result, it can be assumed that changes in the gust related to the stream-wise positioning are not independent of the frequency.

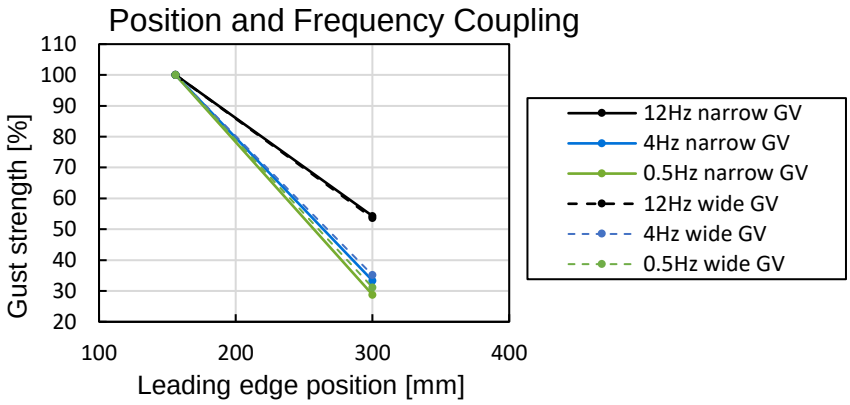


Figure 38: Position and frequency coupling

Considering the results above it was decided to run the subsequent simulation only at one frequency and thus discard the frequency-related differences in gust decay. As the maximum gust angle is one parameter of the optimisation, the optimal gust vane position will tend towards the test section. This tendency is the same for all frequency and is only stronger for lower frequencies. Thus, it can therefore be assumed that disregarding this dependency on frequency will not affect the outcome of the optimisation significantly and therefore justifies a drastically lower simulation effort.

5.2.2 Final Optimisation

All design points excluding the ones already included in the process described in section 5.2.1 were simulated at this point. All these simulations were performed at 12Hz and simulated a sin gust. To evaluate the simulation $v_x$and v_y were logged at a position of x = 70 mm.

The optimisation performed depended on multiple parameters which are shortly described below. The actual values used for the performed optimisation are listed in table 15.

- **Weight w_1**
  The weight w_1 is used in equation 5.1 and defines the importance given to a maximum gust velocity
- **Weight w_2**

The weight w_2 is used in equation 5.1 and defines the importance given to a uniform gust. It is used in combination with the average mean spatial deviation of the gust velocity calculated according to equation 5.3. The deviation is only based on the gust velocity in the area of interest.

- **Area of interest**
  The Area of interest was defined as an area around y = 0 mm in which the test wing inside the test section can move. This distance is based on the movement restriction given to the test wing by the existing test section[5]. To ensure that the test wing in the test section is never exposed to the wake of one of the gust vanes, it must be ensured that the area of interest cannot be reached by a wake at any time.
- **Minimal distance of the gust vane trailing edge to the test section**
  To ensure that the circulation around the gust vane does not interact in any unwanted way with the test wing, a minimal streamwise distance between the trailing edge of the gust vanes and the test wing was defined. This minimal distance was defined based on the separation zone of the gust vanes at maximum deflection. It shall ensure that this zone does not reach beyond the gust generator. The estimated distance was half the gust vane chord length (see fig. 39).

Parameter	Value
w_1	0.2
w_2	0.8
Area of interest	±30 mm
Trailing edge distance	40 mm

Table 15: Optimisation parameter

The weights were a pure design choice as it can be assumed that a given experiment can be adapted to work with lower gust velocities. On the other hand, there is no practical way to adapt it to a gust of low uniformity. Therefore, it was decided to weight the gust deviation much higher than the maximum gust velocity.

The area of interest was calculated as follows:

$$y_{interest} = \Delta y_{test\_wing} + \frac{test\_wing\_thickness}{2} + buffer \qquad (5.4)$$

The buffer was chosen to be equal to half the thickness of the test wing.

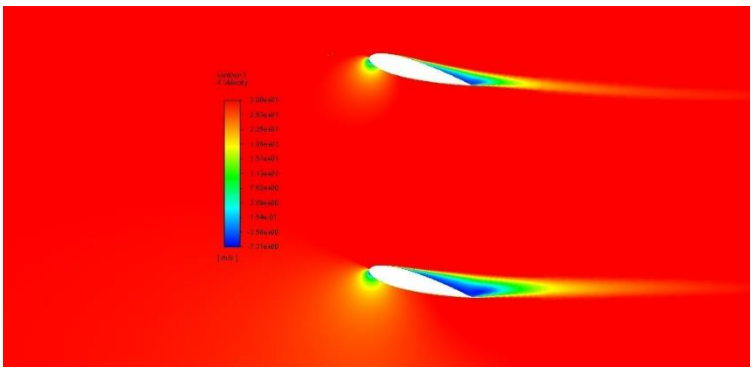


Figure 39: Flow separation, vx cropped 30 m/s; -7.3 m/s

The data gathered with the fluid simulations was post processed with a MATLAB script in multiple steps as described below, for further details see appendix B:

1) In an initial step all simulation files must be stored in a certain folder structure so that the script can loop through all data files.

The following steps are repeated for each design point

Data Collection

2) The file containing the v_y data at x = 70 mm and y = 0 mm for each time step of the simulation is opened and its data is written to a matrix.
3) The data is used to find the time steps where the gust reached its negative and positive maxima for the last time.
4) The file containing various flow data at x = 70 mm across the virtual test section corresponding to the two time steps found in 3) is opened and its data is written to a matrix.
5) The data of the two matrices is split up and saved to different matrices for v_y, v_x and gust angle.
6) The data of each matrix of step 5) is added to a corresponding master matrix to collect the data over all design points.

The following steps are only performed once

7) Various physical values such as maximum gust velocity v_y at x = 70 mm and y = 0 mm or average v_x at x = 70 mm and y = 0 mm over all or design points are calculated and printed to the console.
8) The y – coordinate as well as the v_x value of the wakes generated by the gust vanes at maximum deflection at all design points are collected and stored in corresponding master matrices.
9) Wake-related values are printed to the console

Data Interpolation

10) The master matrices of 6) are copied and the data in the wake regions is replaced with linearly interpolated values.
11) The smoothened data of step 10) is used to interpolate the data in y -direction. At every streamwise (x -) position, the data related to different gust vane spacings is interpolated, resulting in additional design points in relation to gust vane spacing but the still the same number of design points with regard to streamwise position

12) The data of step 11 is used to interpolate the data in x – direction, similar procedure as described in step 11.
13) As the data needed to be re-arranged multiple times during step 11) and 12), the data is now brought back to the form it had after step 10). At this point the data is complete including all iterated design points.

Optimisation Parameter Calculation

14) The data contained in the interpolated matrices is cropped to the area of interest.
15) Average maximum gust velocities are calculated, meaning the absolute gust velocity in the area of interest is averaged including the maximum positive and maximum negative gust velocities. The same is done for the gust angle.
16) The mean maximum gust velocity deviation is calculated for both positive and negative gust velocities and then these are averaged. The mean maximum deviation is then normalized with the value from step 15). The same is done for the gust angle
17) Various values are plotted.
18) The values (only simulated design points) for maximum gust angle and maximum gust velocity at y = 0 as well as the minimum distance of the wake to y = 0 are each fitted to a surface to generate the values for the interpolated design points. These values are evaluated at y = 0 mm.
19) All data for data points which are either violating the restriction given by the minimum wake clearance or the minimum distance of the trailing edge are cropped (set to 0).
20) The remaining maximum gust velocity and gust angle data, as well as the mean deviation values for gust velocity as well as gust angle are normalized in a way that the worst value is represented by 0 and the best by 1.
21) The optimisation parameter according to equation x is calculated.
22) The optimisation parameter as well as the underlying parameters are plotted.
23) The data related to the highest optimisation parameter is printed to the console.

Data Validation

24) The gust velocity as well as the gust angle data at the optimal design point is compared to a simulation done at that point for validation, the interpolated data of the optimisation procedure as well as the simulation data at the same design point is plotted.

In step 10) the wake is removed from the velocity and velocity angle data. This needed to be done as the interpolating schemes available in MATLAB otherwise led to the generation of four wakes in the interpolated data, as the position of the wake changes in between two simulated points. However, this is acceptable, as the data used for the final simulation is only the one in the defined area of interest. The area of interest, by definition, always lies in between the wakes. If this was not the case the data point was cropped as it is not in accordance with the requirements. As data even outside the valid design point range can influence the interpolation in that range the process was done in the order described above: First interpolation and then cropping of the data. In step 18) it is described that certain values were interpolated directly from the simulated data and were not calculated from the interpolated data points of step 11) and 12). This could be done as they are only reliable on data at y = 0 ant therefore not affected by any wake. Thus, all subsequent steps were not needed for the interpolation of these values and were left out to have a more streamlined calculation.

5.2.3 Optimisation Results

The spatial gust profiles at the time of maximum gust amplitude display an interesting behaviour. It can be clearly seen that the maximum gust angle is increasing if the gust vanes are closer to the test section and thus to the measurement point as well as when the spacing between the gust vanes is smaller (see fig. 40 -43). Both trends were already described for the gust generator build for the OJF [4]. As mentioned earlier, the flow inside the gust generator is not parallel due to the narrowing cross section. The influence of this shape on the gust can be seen clearly. For configurations with the gust vanes positioned farthest away from the test section the gust is subjected to flow toward the centre. Figure 40 represents the gust velocity corresponding to a maximum distance between test section and gust vanes. The gust is in that case subjected to downflow of increasing strength towards the upper wall of the gust generator. Therefore, the positive gust velocity gets cancelled out by the downflow and gets smaller towards the wall.

If the gust vanes are positioned closer to the test section, they automatically move closer to the gust generator walls (see fig. 36). If the spacing in between the gust vane is increased as well, they move close enough to the wall to create a flow blockage, which can clearly be seen in the pressure distribution in figure 44. This distorts the flow in such a way that the gust is not subjected to the before mentioned downflow after the gust vanes. It seems as if the downflow rather passes through the gust vanes, supported by the flow blockage between the gust vane and the gust generator wall, and thus mostly affects the gust velocity at positions furthest away from the blockage. As a result, the gust profile shows now highest gust velocity in an area close to the upper gust vane (see fig. 43).

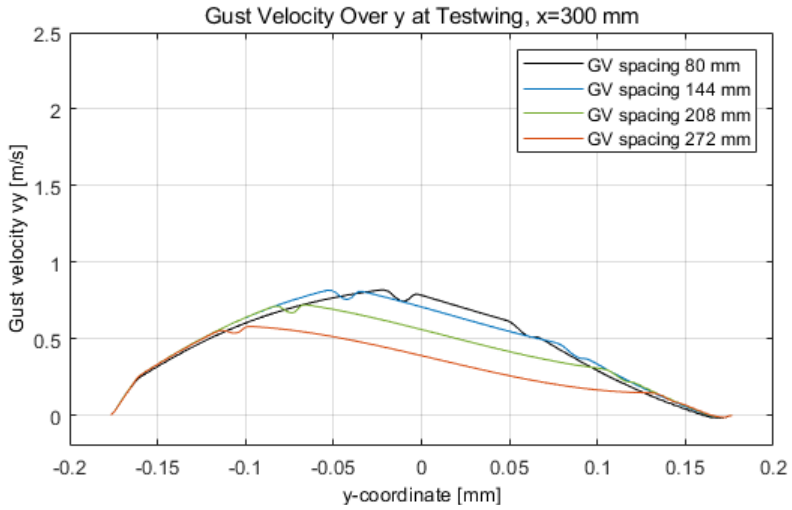


Figure 40: Gust velocity at max. amplitude for x= -300 mm

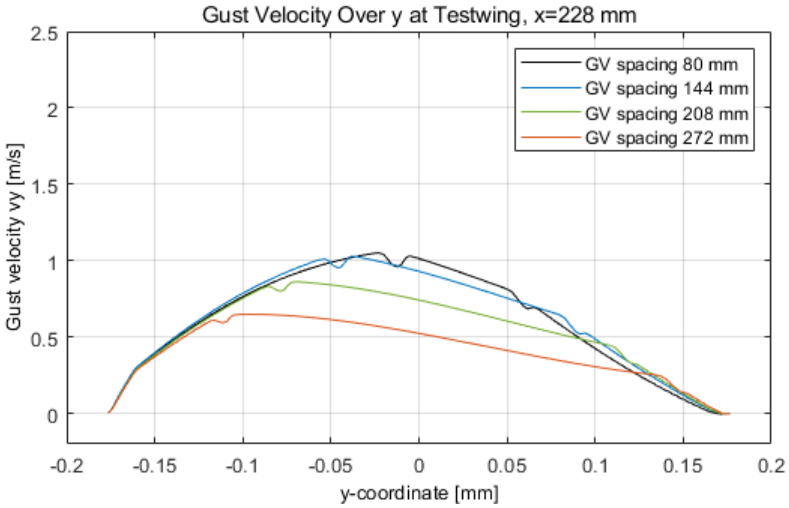


Figure 41: Gust velocity at max. amplitude for x= -228 mm

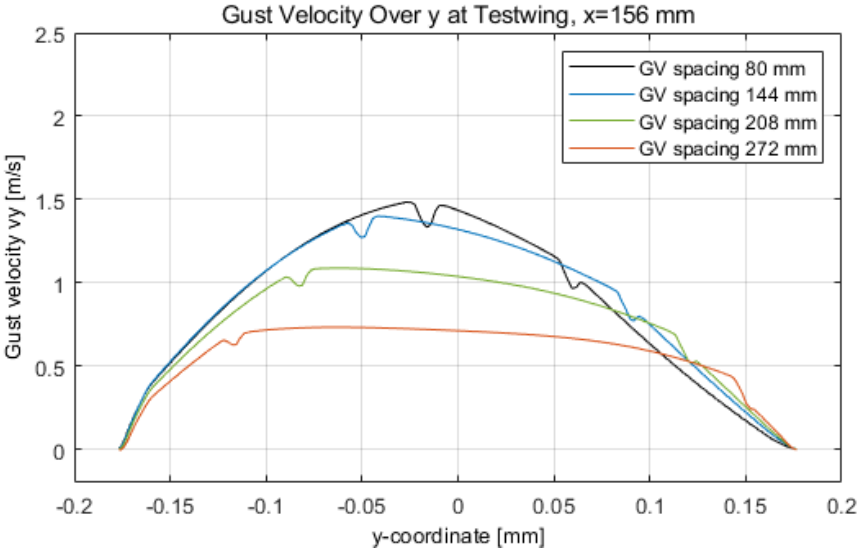


Figure 42: Gust velocity at max. amplitude for x= -156 mm

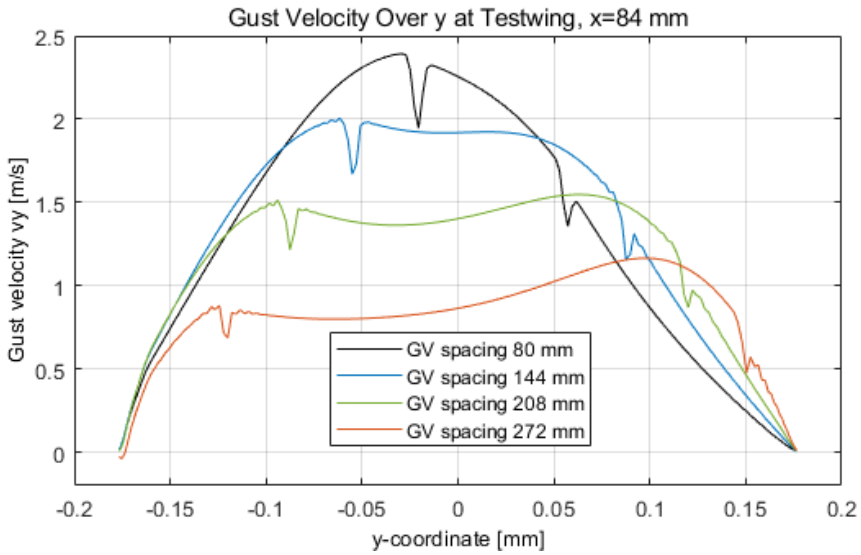


Figure 43: Gust velocity at max. amplitude for x= -84 mm

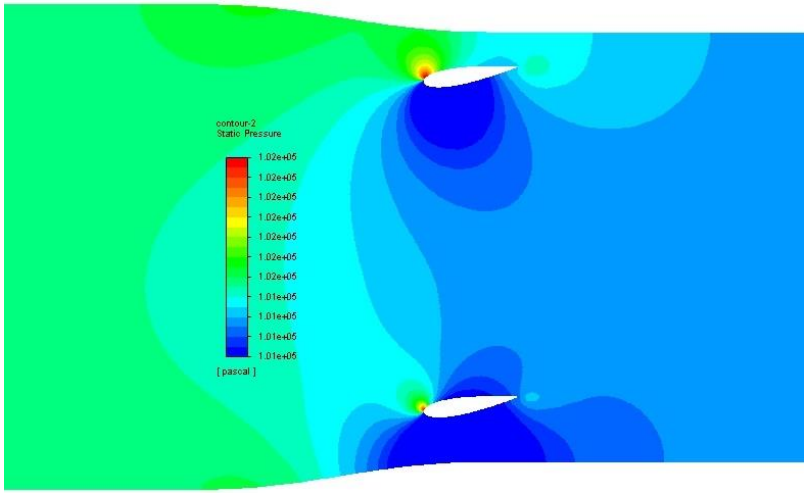


Figure 44: Pressure distribution for gust vanes in wall proximity

The optimisation delivered results in accordance with the above described flow characteristics. Figure 45 and 46 illustrate the two quantities v_{ymax} and d_{vymax}. All data points which did not meet the minimal wake clearance were set to 0. All data points where the gust vanes were too close to the gust generator exit and therefore violating this condition are not shown. One can clearly see that the maximum gust angle is tending to bigger values for gust vanes closer together and further downstream. It can be observed that the deviation is at a similar low level in a region where an in setting blockage starts to correct the distorted gust due to the downflow after the gust vanes. It has to be noted that the deviation was only calculated in the area of interest of 30 mm > y > -30 mm. The values of the final optimisation parameter calculated according to equation 5.1 can be seen in figure 47, with the optimum encircled in red. The optimum indicated a gust vane spacing of 160 mm. As the gust vanes could touch in this configuration during maintenance mode (see section 3.2), it was decided to use the next design point with a spacing of 168 mm. This has no significant effect on the gust velocity deviation but decreases the maximum gust velocity slightly. This was however accepted in favour of a better usability. Said design point is encircled in green. The final numerical results of the optimisation can be found in table 16.

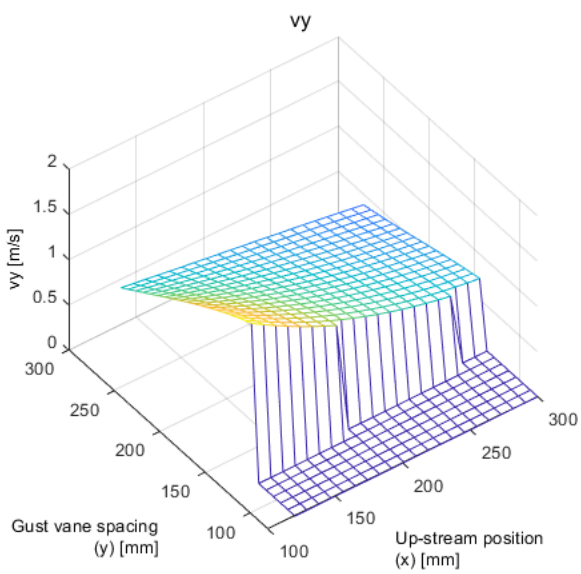


Figure 45: Maximum vy for all design points

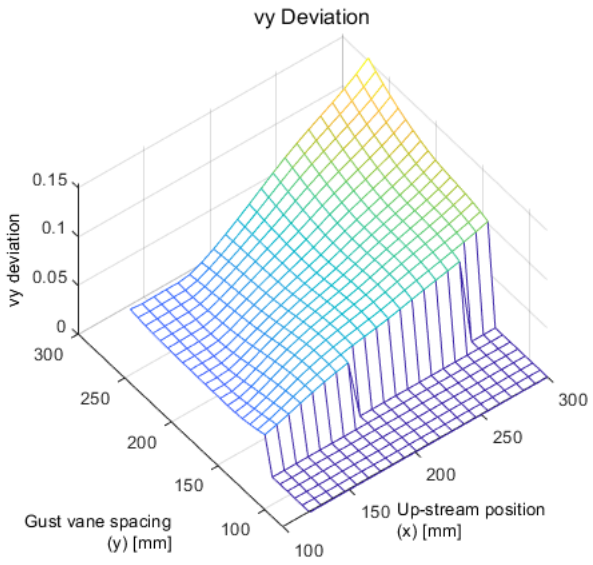


Figure 46:Mean vy deviation for all design points

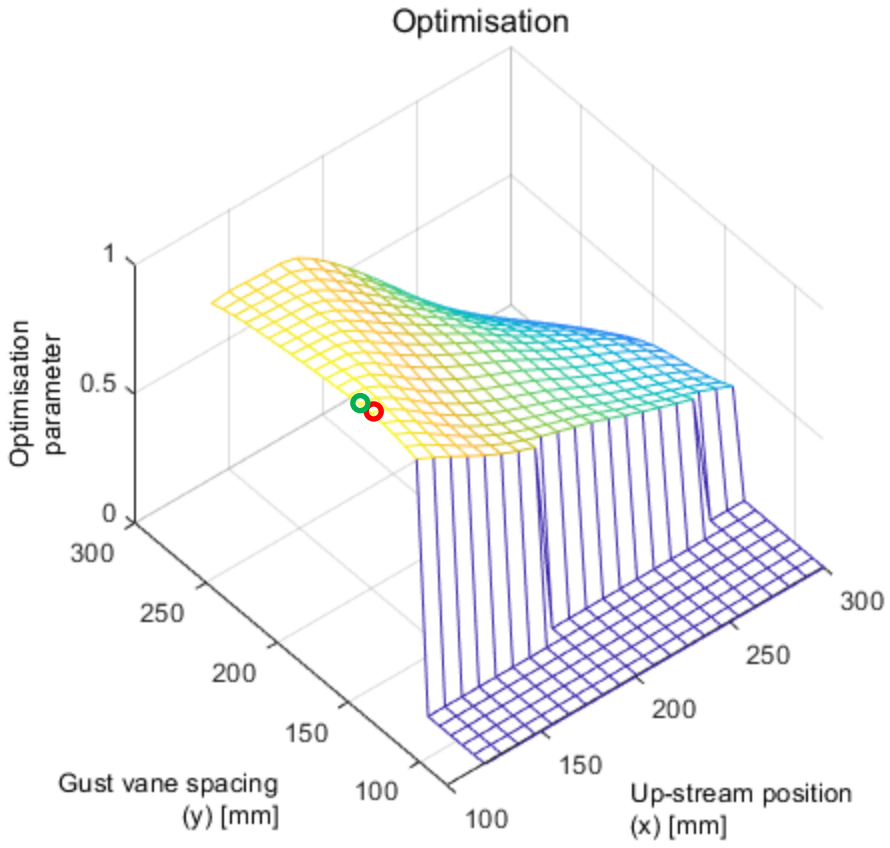


Figure 47: Optimisation parameter for all design points

As these values are based on interpolated data, a flow simulation was performed at this design point to validate the result. As it can be seen in figure 48, The curves are not exactly the same, but were considered to be matching well enough in the area of interest. The simulated values of v_{ymax} and d_{vymax} are mentioned in table 16 as well.

Data Source	v_{ymax} [m/s]	d_{vymax} [%]	Design Point
Interpolated design point as optimized	1.533	3.03	120 mm ,160 mm
Interpolated point as chosen	1.487	3.01	120 mm, 168 mm
Simulated design point as chosen	1.476	3.42	120 mm, 168 mm

Table 16: Optimisation results

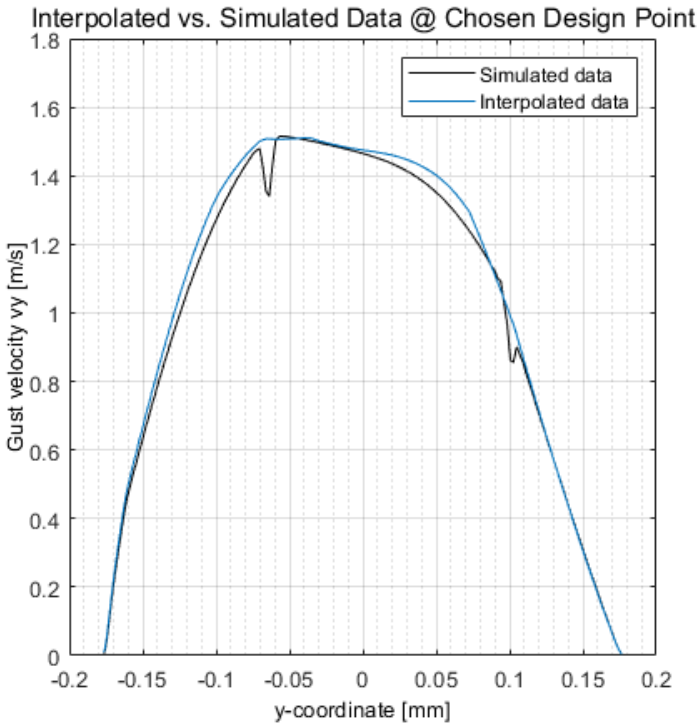


Figure 48: Gust shape of interpolated data vs gust shape of simulated data

6 Design

After the position of the gust vanes was determined by the optimisation process described in section 5, the gust generator could be designed in detail. At first the hardware components necessary to actuate the gust vanes needed to be selected. Subsequently the complete system could be designed with CATIA V5.

6.1 Preliminary Calculations

Preliminary calculations were done to generate an initial estimation of the aerodynamical and mechanical loads which are to be expected during the operation of the gust generator.

As the data generated by XFoil predicted a similar maximum Cl value as the CFD simulations, the more conservative estimation of XFoil was used (see fig. 49). Regarding this data, the maximum Cl value was determined to be 1.2 and is reached roughly at ±15°, which was earlier defined to be the motion limit for the gust vanes.

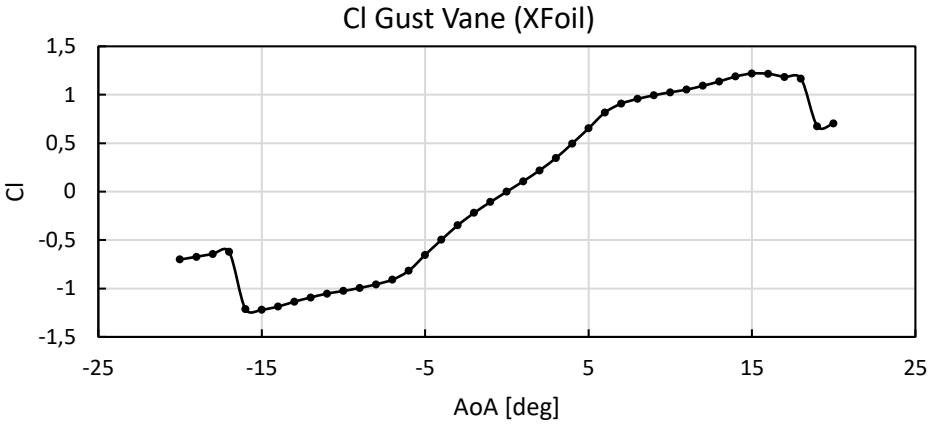


Figure 49: Cl of Gust vane calculated with XFoil

In order to calculate the torque needed to accelerate and decelerate the gust vanes, the moment of inertia was determined using a model of the gust vane in CATIA V5. All values need for the preliminary calculations were known at this point and are listed in table 17:

Parameter	**Value**
Fluid density	1.225 kg/m^2
Fluid velocity	30 m/s
GV chord length	0.08 m
GV span (= adapter section depth)	0.4 m
Maximum coefficient of lift of GV	1.2
GV moment of inertia	5.32*10^{-4} kg*m^2
Maximum GV angle	15°
Maximum Motion Frequency	12 Hz

Table 17: Preliminary calculation parameter

All calculations were done with the common sign convention as shown in figure 50. Note that the angle

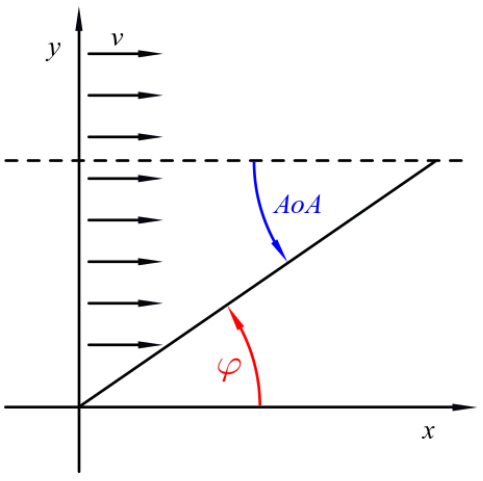


Figure 50: Sign convention

of attack (AoA) is measured in clockwise direction while global angles (meaning all other angles) are measured in counterclockwise direction. Therefore, if the gust vane angle φ is positive, the gust vane is at a negative angle of attack.

To calculate the aerodynamical moment on the wing, the maximum lift is calculated as

$$L = \frac{1}{2} c_l \rho v^2 cs \tag{6.1}$$

where c_l is the lift coefficient, ρ the density of the fluid (air), v the flow velocity, c the chord length and s the gust vane span. By plugging in the values of table 17 and the values stated in section 4.2 in table 4, a maximum lift of $L = 21.17\ N$ is obtained. As the gust vane is a symmetric profile, its centre of pressure lies roughly at $0.25c$ and does not move with a change in angle of attack. A pivoting point between $0c$ and $0.5c$ is considered to be reasonable for the gust vanes. Therefore, the maximum aero dynamical moment is calculated for the two extreme positions of the pivoting point as follows:

$$M_{aero} = \pm 0.25c * L \tag{6.2}$$

This leads to a maximum aerodynamical moment for a negative angle of attack/a positive gust vane angle of either $-0.424\ Nm$ if the pivoting point is at the leading edge or $0.424\ Nm$ if the pivoting point is at half-chord.

Based on the two gust types described in section 3.1, the motion of the gust vane for a 1 -cos gust was defined as

$$\varphi = \frac{\varphi_{max}}{2}(1 - \cos 2\pi f t) \tag{6.3}$$

Whereas the gust vane motion for a sin gust was defined as

$$\varphi = \varphi_{max} * \sin(2\pi f t) \tag{6.4}$$

where φ is the gust vane angle and φ_{max} the maximum gust vane angle. Note that a positive angle of attack is represented by a negative angle of φ, as φ is defined to be the angle between the x-axis and the chord of the vane with the positive direction as seen in figure 50. It can be shown that the maximal acceleration for the sin gust is bigger than the one for the 1-cos gust with the same φ_{max}. The following calculations are therefore performed for the sin gust. Differentiating equation 6.4 with regard to t leads to

$$\omega = \varphi_{max} * 2\pi f * \cos(2\pi f t) \tag{6.5}$$

where ω is the angular velocity. Differentiating equation 6.4 a second time leads to

$$\dot{\omega} = \varphi_{max} * (2\pi f)^2 * -\sin(2\pi f t) \tag{6.6}$$

where $\dot{\omega}$ ist he angular acceleration the maximal angular velocity and acceleration can easily be derived from equation 6.5 and 6.6:

$$|\omega_{max}| = \varphi_{max} * 2\pi f \tag{6.7}$$

$$|\dot{\omega}_{max}| = \varphi_{max} * (2\pi f)^2 \quad (6.8)$$

Using the values from table 17, we get $|\omega_{max}| = 19.739 \frac{rad}{s} \rightarrow 188.5\, rpm$ and $|\dot{\omega}_{max}| = 1488.3 \frac{rad}{s^2}$. The velocity magnitude is reached when the gust vane passes 0°, whereas the maximum acceleration is reached when the gust vane is at its maximal deflection (see fig. 51).

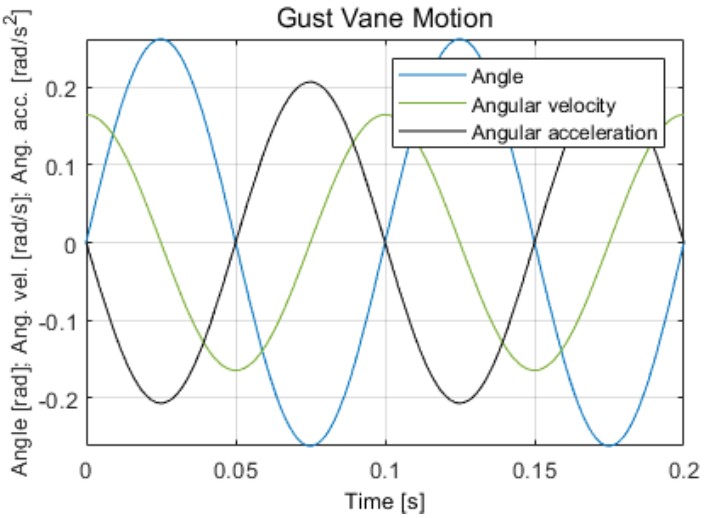


Figure 51: Gust vane motion, angular vel. scaled by factor of 1/100 and angular velocity by 1/5000

The mechanical torque generated by the inertia of the gust vane can then be calculated as

$$T_{mech} = -(I * \dot{\omega}) \quad (6.9)$$

where I is the moment of inertia of the gust vane. The mechanical torque corresponds to the torque to which an actuator would be subjected to while moving the gust vane, not considering the aerodynamic load.

With the maximal angular acceleration calculated above and the moment of inertia of the gust vane a maximum mechanical torque of $0.792\, Nm$ is derived at a positive gust vane angle. This calculation does not yet consider any mounting parts to connect the gust vane to the actuator. These would increase the Inertia of the gust vane assembly ant thus also increase the torque required to move it.

The complete load that is generated by the gust vane can then be derived by simply adding the maximum aero dynamical moment and the maximum mechanical torque:

$$T_{maxload} = M_{maxaero} + T_{maxmech} \quad (6.10)$$

With the values above this leads either to $0.368\, Nm$ if the pivoting point is at the leading edge or to $1.216\, Nm$ if the pivoting point is at half-chord (for a negative angle of attack).

The power needed to move the gust vanes can thus be estimated as

$$P_{max} = T_{maxload} * \ |\omega_{max}| \quad (6.11)$$

This calculation ignores the moment of inertia of all other involved parts as well as of the actuator itself. Additionally, the aerodynamic load was calculated assuming steady aerodynamics. The power calculated with equation 11 is over estimated as the maximum torque is delayed by 90° with regard to the

maximum velocity. However, as the goal of the preliminary calculations were to estimate the torque and power requirement this is considered to be sufficiently accurate at this stage. The final results of the preliminary calculations can be found in table 18.

Parameter	Value
$\lvert\omega_{max}\rvert$ [rad/s]; rpm	19.739; 188.5
$\lvert\dot{\omega}_{max}\rvert$ [rad/s$^{2]}$	1488.3
$\lvert T_{maxload}\rvert$ [Nm]	1.216
P_{max} [W]	24

Table 18: Preliminary calculated values

6.2 Hardware

Based on the problem description in section 3.1, and the preliminary calculations performed in section 6.1 a set of specifications which must be met by the hardware components were defined. Based on these specification different options were generated and a final decision was made. This process was performed for the actuator group, consisting of the combination of actuator and gearbox, as well as for the electronics used to control the system.

6.2.1 Specifications

The specification can be grouped in the three categories mechanical, functional and usability.

- **Mechanical**

  The mechanical specifications are directly based on the preliminary calculations described in section 6.1. Additionally, to the already established specifications for torque and rotational velocity, the so-called inertia ratio was taken into account. Dynamic behaviour of a mechanical system is linked to this parameter. To ensure appropriate dynamic behaviour, the inertia ratio is supposed be below 5 and not exceed 10 if a servo motor is used. For a stepper motor an inertia ratio of 1 is not to be exceeded [31]. The inertia ratio is defined as

$$R = \frac{I_R}{I_A} \tag{6.12}$$

  where I_A is the inertia of the actuator itself and I_R is the reflected inertia. The reflected inertia represents the inertia of all components that must be driven by the actuator as it is experienced by the actuator itself. The total torque, which needs to be generated by the actuator is given as

$$T_{maxactuator} = -(\frac{M_{maxaero}}{GR} + \frac{(I_{GV} + I_M) * -\dot{\omega}_{maxL}}{GR} + (I_G + I_A) * -\dot{\omega}_{maxL} * GR) \tag{6.13}$$

  where $M_{maxaero}$ is the aerodynamic moment, I_{GV} the moment of inertia of the gust vane, I_M moment of inertia of the mounting of the gust vane, I_G themoment of inertia of the gearbox (with respect to the input), I_A the moment of inertia of the actuator, GR the gearing ratio, and $\dot{\omega}_{maxL}$ the maximum angular acceleration of gust vane. With

$$\dot{\omega}_{maxM} = \dot{\omega}_{maxL} * GR \tag{6.14}$$

where $\dot{\omega}_{maxM}$ is the maximum angular acceleration of the actuator, one can rearrange equation 6.13 to:

$$T_{maxactuator} = -\frac{M_{maxaero}}{GR} + \left(\frac{I_{GV} + I_M}{GR^2} + I_G + I_A\right) * \dot{\omega}_{maxM} \quad (6.15)$$

Finally, the reflected inertia I_R is then defined as

$$I_R = \frac{I_{GV} + I_M}{GR^2} + I_G \quad (6.16)$$

At this stage, many variables of equation 6.15 are still unknown and the initial selection for a suitable actuator was based on the estimated required power of 24 W calculated in section 6.1. Based on this assumption, actuators with a power output between 50 W and 100 W were considered to be reasonable. The inertia ratio was calculated using the data of the potential actuators and the I_{GV} only. A safety margin of at least 1.5 with regard to output torque was considered to be reasonable keeping in mind that the maximum required torque is already overestimated by just adding up the aerodynamic and the dynamic load. Further, in accordance with the requirements described in section 3.3 with regard to system stiffness, it was defined that the gearbox must show a high stiffness of above 0.5 Nm/arcmin as well as a low backlash below 15 arcmin.

The mechanical specifications are summarised in table 19.

Parameter	Value
Min torque	1.216
Min. rotational velocity [rad/s]; [rpm]	19.739; 188.5
Min. Power [W]	24
Max. Inertia Ratio Servo/Stepper motor	5; 1
Min Gearbox Stiffness	0.5 Nm/arcmin
Max. backlash	15 arcmin

Table 19: Mechanical specifications

- **Functional**

The specifications for the functionalities which the involved hardware must have are given below. They are derived on the functional requirements for the gust generator as described in section 3.1

- The control hardware and the actuators must be compatible
- The control hardware must be programmable in a flexible manner
- The control hardware must be small enough to fit in an enclosed system on top of the adapter section
- A laptop or desktop must be able to connect over a network or physically to the control hardware to provide input parameter for the motion control
- The actuators must be able to perform at least position controlled moves. Therefore, low level motor control including sensors must either be embedded in the actuator or must be available for them

- **Usability**
  The specifications for usability are not directly related to the performance of the system but facilitate its use. They are therefore considered to be highly desirable but are not necessary for the system to work.
  - All hardware components are suggested to be used together/are offered as a package
  - Technical support is available
  - Modular structure which allow for changes to improve the system or change its capabilities

6.2.2 Actuation Concepts

Three basic actuating concepts were considered as an option for a gust generator in the present size range.

- **Small servo motors as used in RC models, directly controlled by a single board computer with pulse-width modulation:**
  This concept is the easiest to implement, as various single board computer (SBC) have libraries available to control RC model servos with pulse-width modulation (PWM). However, with a typical power output of only roughly 1.5 W and possible angular velocities of about 5 rad/s they are simply mechanically not powerful enough for the task at hand.
- **Stepper motors with low-level control by a stepper driver, and high-level control by an SBC**
  Using a stepper motor has some advantages particularly for positioning tasks: It provides high holding torque and even detent torque when the power is cut off. The precision is given by the incremental step size defined by the motor design itself. However, the task at hand requires continuous dynamic motion with precise movements.
- **Servos/integrated servos, low-level control by servo drive/integrated servo drive, high-level control by plc or SBC**
  Servo motors are available in different configurations. As simple motors combined with some sort of sensor to control the speed and/or the position of the motor axle and the necessary power electronics. Others have an integrated motor drive or even a motor drive and a motor controller. The precision is limited only by the resolution of the sensors used in combination with the servos. In principle all variants are usable, depending on which form of communication is desired to use (with limitations in communication speed). However more integrated solutions are preferred due to easier handling and a more compact design. The high-level control can in most cases be done with either an SBC or a programmable logic controller (PLC).

The last concept was chosen to be used for the development task at hand. Servo motors are better suited as stepper motors as they show less vibration, are quieter and are not bound to incremental steps. The task at hand would be possible with stepper motors, but a solution with servo motors is preferred [32]. The top-level control will be performed by an SBC as it is a small computer and therefore offers the desired high flexibility with regard to control concepts. This flexibility allows for future changes, optimisations or extensions of the control software.

6.2.3 Selected Components

Based on the specifications defined in section 6.2.1 and the concept of section 6.2.2 suitable hardware options were searched. The low-level control was hereby treated as part of the search for a suitable servo as it was considered to be necessary that these components are either integrated in the servo or are provided as a package with the servo. Table 20 shows the SBC's considered for the high level control of the system whereas table 21 shows the considered servo drives.

Parameter	BeagleBone Green	BeagleBone Enhanced, industrial	Raspberry Pi 4
Connection over network possible?	Yes	Yes	Yes
Ethernet	10/100	GB	GB
USB host	1	1	1 (power only)
USB client	1	2+2	4 (2X USB 3)
Other ports	UART/c2c, 2*46 Pin	2*46 Pin	Multiple “media” ports; 40 Pin
Processor	Broadcom BCM2711	AM335x ARM®	AM335x ARM®
Built-in sensors	No	Gyro/Baro/Temp	No
Expandability	Yes	Yes	Yes

Table 20: Single board computer options

Parameter	Teknic ClearPath SCSK 2310P/2310S	Simplex Motion SC020B/SM100A	Applied Motion J0100-303-3-000[7]	JVL MAC095 w. MAC00-B1 module
Continuous power out put	66/100 W	50/100W	100 W	92 W
Continuous torque	0.155/0.31 Nm	0.12/0.32 Nm	0.32 Nm	0.22 Nm
Peak torque	0.791/1.575 Nm	0.4/2 Nm	0.96 Nm	0.62 Nm
ω_{max}	4000 rpm; 419 rad/s	3000/4000 rpm; 314/419 rad/s	2900 rpm; 303 rad/s	4000 rpm
Rotor inertia	$0.1*10^{-4}$ kg*m^2	0.126/0.78 10^{-4} kg*m^2	$0.043*10^{-4}$ kg*m^2	$0.119*10^{-4}$ kg*m^2
Gr_{min} (torque based)	7.85/3.92	10.13/3.8	3.8	5.5
Gr_{opt} (inertia ratio based)	7.3	6.5/2.6	11.2	6.7
Proposed Gr	15/10	15/10	10	10
Safety factor (torque)	1.9/2.5	1.48/2.63	2.63	1.8
Resolution	0.45 (0.057[8]) deg. (absolute)	0.09 deg. (Absolut)	0.144 (incremental)	0.35 (incremental)
Communication	Over Hub: USB RS232	Over Hub: USB RS485 Direct: RS485/232 TTL USB[9], AS	Over Drive: RS485 AS	Direct: RS232/422/485 AS
Signal/Control modes	S/D, Software API	S/D, AS, Modbus, quad. encoder input, custom digital control	S/D, AS, streaming com-mands	S/D, AS, custom digital control
Control integration	Sensor/Drive/MC	Sensor/Drive/MC	Sensor	Sensor/Drive/MC

Table 21: Servo options; S/D = step and direction, AS = analogue signal

[7] With SV2D10-Q-RE motor drive
[8] With advanced option
[9] Direct USB only for SM100A

The decision was made in favour of the BeagleBone green in combination with Teknic ClearPath SCSK 2310S servos. The BeagleBone green is based on the BeagleBone black, which was especially designed for IoT and automation applications in collaboration with a third party (seed studio). The BeagleBone green features additional connectors for sensors. BeagleBone offers extension capes which allow to increase the functionality of the SBC at hand. Multiple sensors, ready to use are available for the BeagleBone green. The BeagleBone green features programmable real time units (PRU) which offers useful real time capabilities.

The Teknic ClearPath SCSK 2310S fulfils all technical specifications and is designed ready to use. The higher-powered version is to be chosen, as this leads to a larger safety factor. All additional electronics needed to operate these servos are provided by the manufacturer as well simplifying the development. The ClearPath SCSK servo motors can be operated in the three modes software motion control, by a step and direction signal or by a quadrature A/B signal. This increases the flexibility for the initial development as well as for further improvements or changes. The mode can be set with the ClearView software provided by the manufacturer. Teknic provides a software library called sFoundation as part of a software development kit (SKD) to facilitate the development of the control software. Software developed based on this library can be used in combination with all three operating modes, however it does not include any tools for the signal generation for step and direction or quadrature A/B. High level control for these servos can be done with the BeagleBoneGreen as stated by the manufacturer.

Based on the servo selection, a gearbox with a gearing ratio of 10:1 was necessary (see tab 22). Gearboxes of the manufacturers WITTENSTEIN SE, Wilhelm Vogel GmbH, Harmonic Drive SE, RECKON DRIVES INTERNATIONA and Applied Motion Products, Inc. were considered in the selection process. Special emphasis was given to low backlash as this was reported to be critical with the gust generator for the OJF. All gearboxes fulfilling the specifications stated in section 6.2.1 were of the planetary type. The gearboxes were finally ordered from WITTENSTEIN SE due to time restrictions as lead times of all other products would have over stressed the time frame of this thesis. However, one can say that all viable options had very similar specifications. The final gearbox configuration was derived in collaboration with engineers from WITTENSTEIN SE. For further detail refer to appendix D.

The main characteristics of the gearbox are listed in table 22. An overview of all final hardware components is given in figure 52 and table 23.

Parameter	**Value**
Gearing ratio	10:1
Maximum continuous torque [Nm]	21 Nm
Maximum [rpm]	10’000
Moment of Inertia [kg*m^2]	4*10^{-6}
Torsional stiffness [Nm/arcmin]	0.85
Maximum backlash [arcmin]	10

Table 22: Gearbox specifications

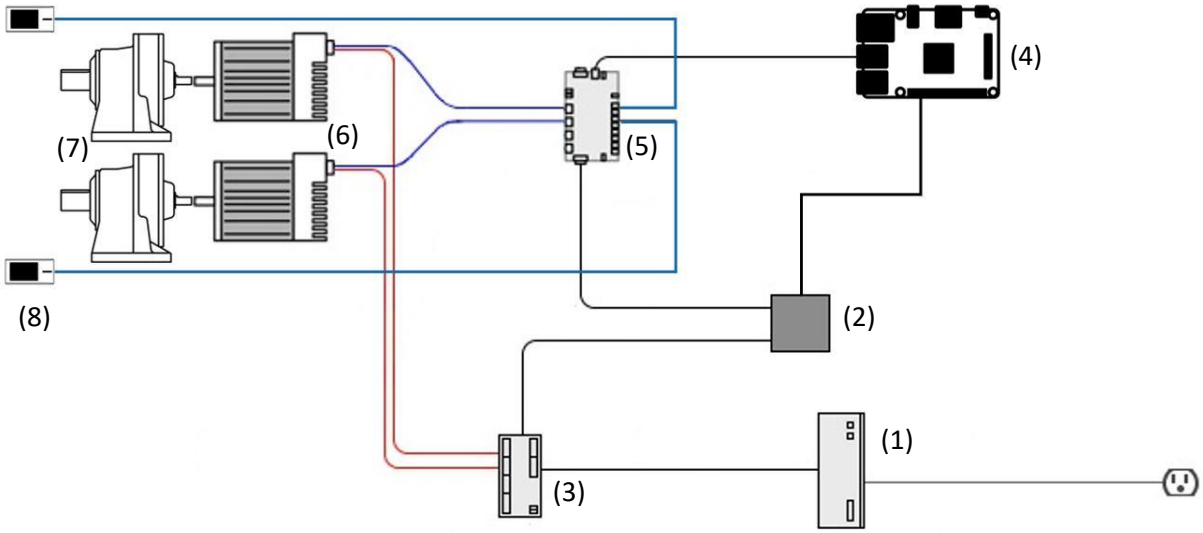


Figure 52: Hardware overview

No	Component	Function
1	Bus Power supply	
2	Logic Power supply	
3	Power Hub	Distributes bus power to actuators
4	BeagleBone Green	High level control
5	Communication Hub	Communication interface between high and low level control
6	Servo Motors	
7	Gearboxes	
8	Hall sensors	Used to limit motion as well as for actuator homing

Table 23: Hardware components

6.3 Mechanical Design

The mechanical design was done with a focus on rigidity and the avoidance of free play. The mounting of the wings was designed to make them removable. The gust vane assembly can be seen in figure 53. The gust vanes (1) were screwed to two end pieces (2). The one on the side of the actuators features a grove (3) for a small magnet to be detected by the hall sensor. These endpieces are designed to slide into circular adapter pieces at each end (4). These pieces fit each into a flange (5) at each end and seal the inside of the gust generator towards the outside. A shaft (6) is connected to each of these circular adapter pieces. On the actuator side three support pieces (7) mount the gearbox in place which is connected to the shaft with a rigid clamp style coupling. On the other side the gust vane is supported by a bearing seated in a bearing housing (8) which is itself connected to the flange (5).

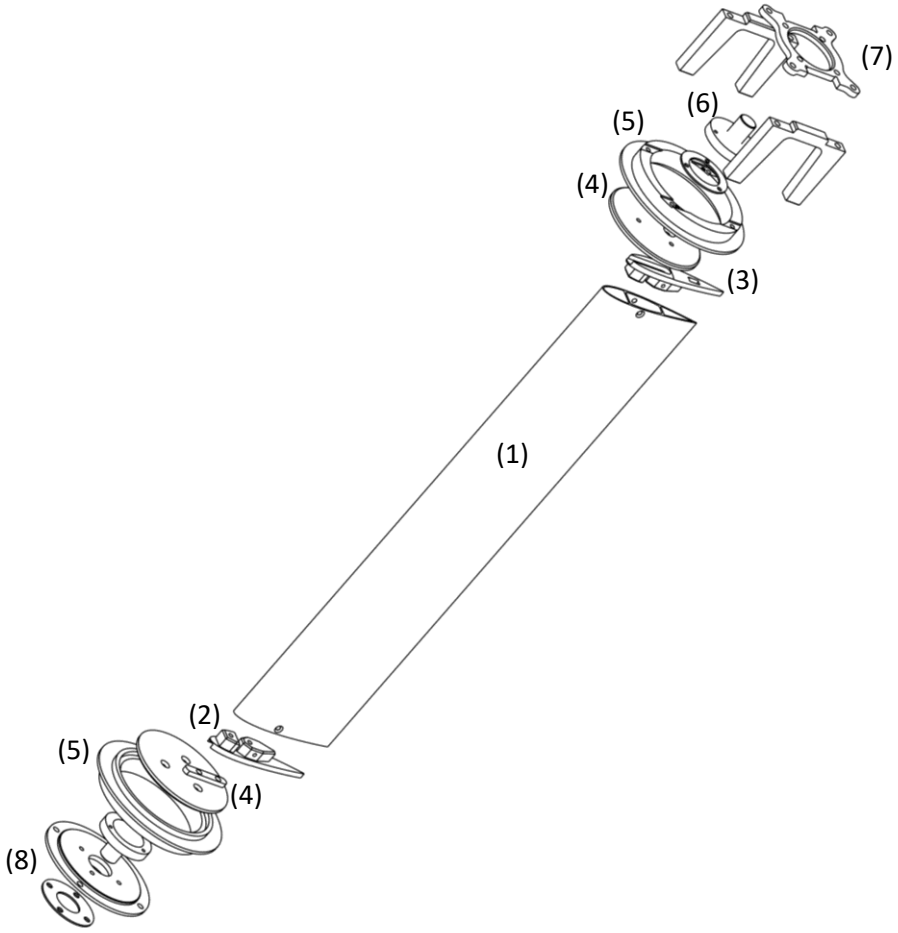


Figure 53: Gust vane assembly

As the wooden adapter section in which the gust vanes are mounted could not be considered to be precise enough, the holes to fit the whole assembly was cut out oversized. Both flanges feature an oversized lip to cover the oversized hole. They were glued in on both sides with epoxy using a laser cut gage to ensure a precise placement (see fig. 54). After the epoxy was fully cured the remaining gaps were filled with wood filler and sanded smooth. Prior to the mechanical assembly, the inside of the gust generator was spray painted black to reduce reflection in case of PIV testing as well as to protect the plywood from any seeded particles or droplets (see fig. 55).

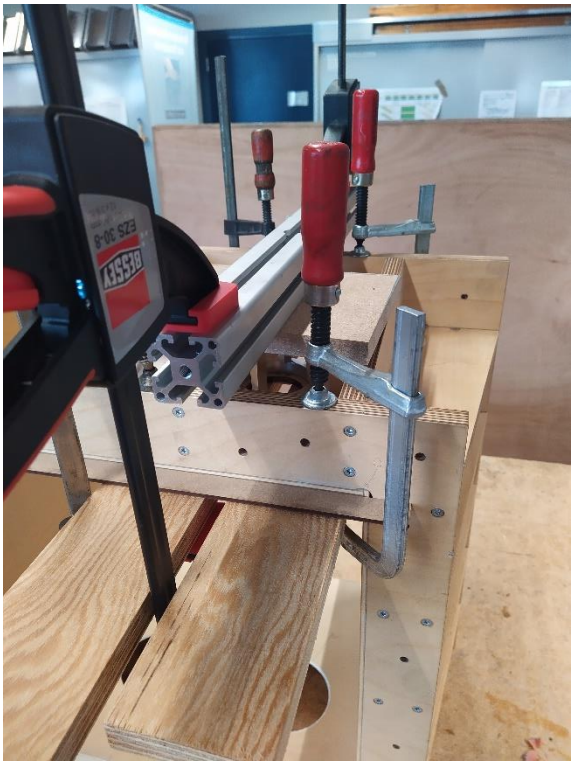


Figure 54: Gluing flanges placed with gage

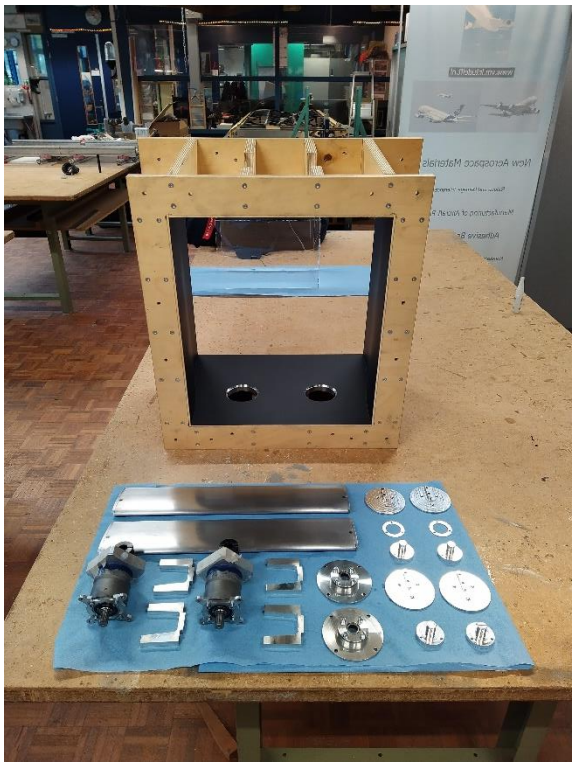


Figure 55: Painted gust generator prior to assembly

The hardware placement is shown in figure 56. Emphasis was put on a fully enclosed system. Therefore, all hardware components including the cabling are placed inside a case made from acrylic glass to ensure any indicator and or warning LED's on the hardware components are visible.

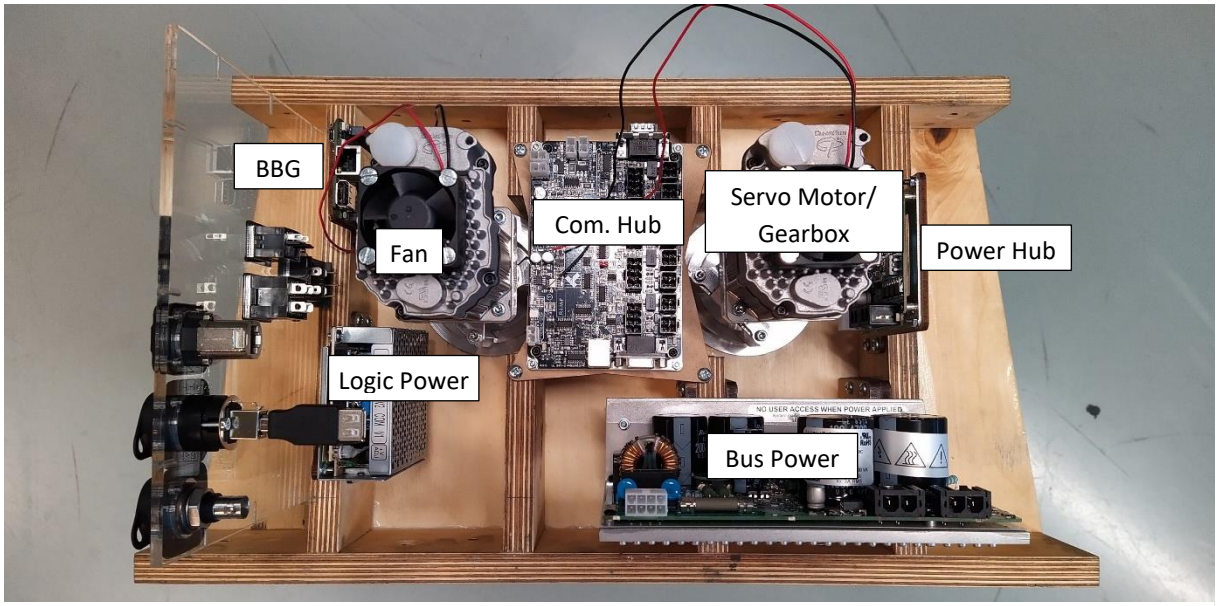


Figure 56: Hardware placement

To allow for communication feedthroughs for ethernet (1), USB (2) and BNC (3) connectors are installed on a panel (see fig. 57). The BNC is reserved as output for the trigger signal further described in (section 7). To protect the single board computer from any harmful voltage, three switches were installed to split the start-up procedure. After the main power is enabled (4), the BeagleBone green can be powered up (5) prior to the rest of the electronics and thus ensuring that its output rail is supplied with voltage before any other component could accidentally feed any harmful signal to the rail. With the last switch (6) all the remaining hardware is powered up.

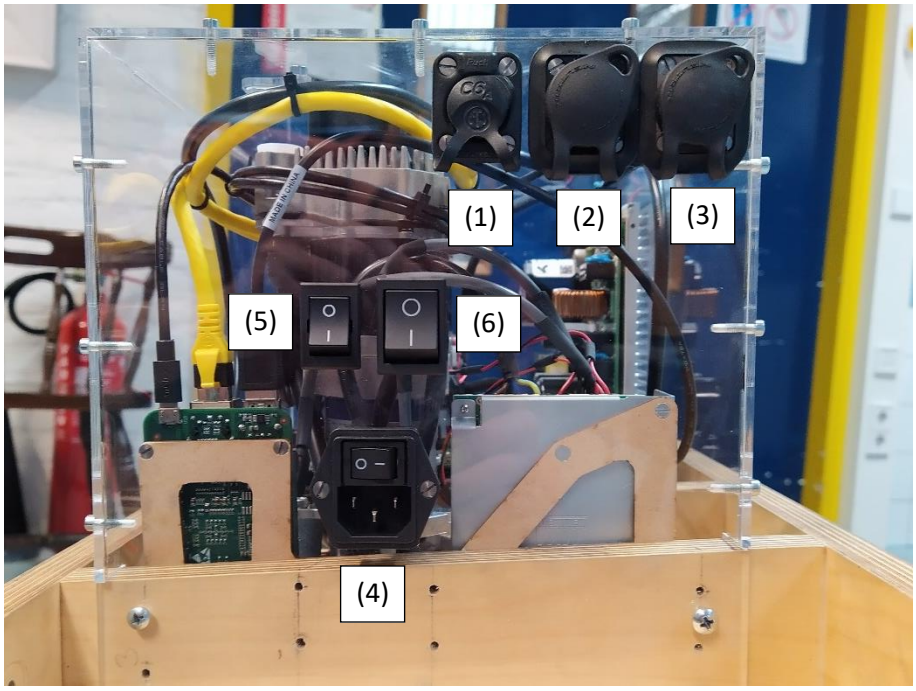


Figure 57: Front panel

The case features openings over the fans to allow for optimal ventilation as well as small hatches to give access to diagnostics ports on the servo motor (see fig. 58). This allows for safe monitoring of the servos when the system is running. Figure 59 shows the completely assembled gust generator attached to the wind tunnel.

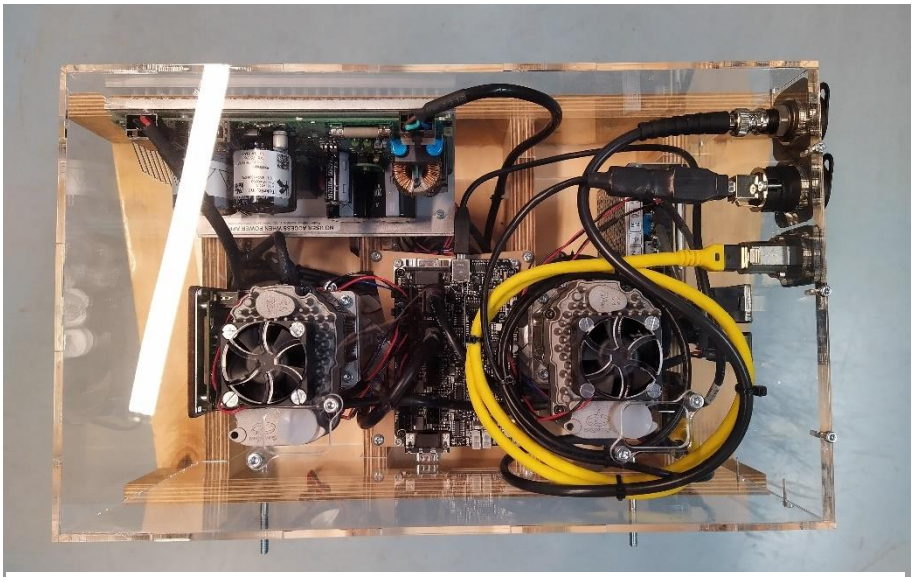


Figure 58: Acrylic case

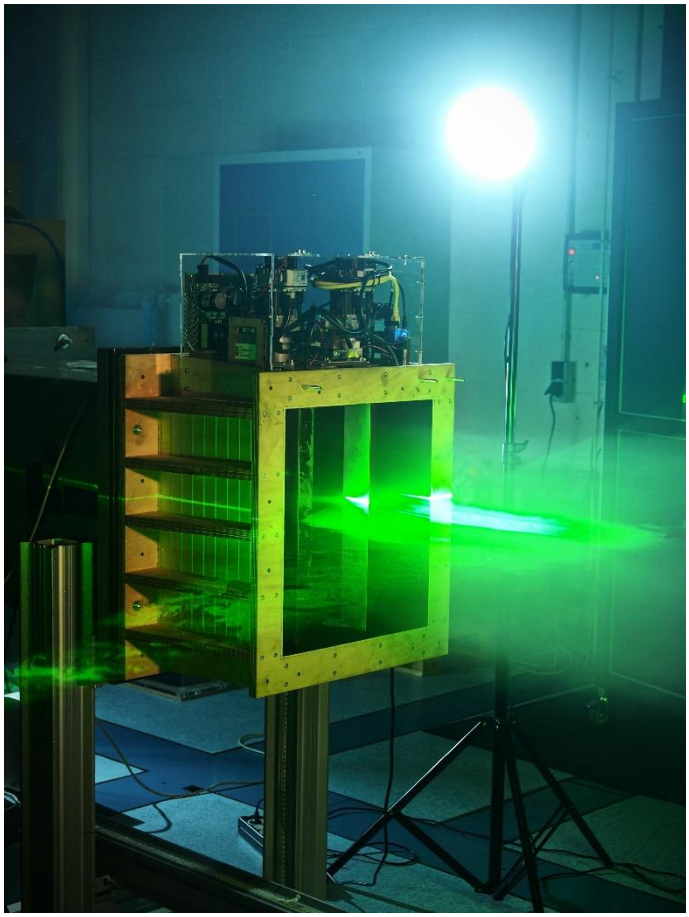


Figure 59: Gust generator

7 Motor Control

The hardware used in the gust generator requires control code in order to operate the servo motors as desired. The following sections describe the chosen control concept and how it was implemented in said control software.

7.1 Motor Configuration and Control Mode

The initial motor set-up is done with ClearView, the configuration software provided by Teknic. The software allows to tune the low-level motor control in the ClearPath SCSK servo motors to the mechanical system. Additionally, one can define a homing procedure as well as the motor behaviour on input signals. A maximum of two input per motor is available. These functionalities were used to configurate the motors to home with the help of the installed hall sensors. The same hall sensors were used as well to implement a soft stop to keep the vanes from exiting a certain range of motion.

As mentioned earlier, the possibilities of the sFoundation library are limited if the servo motors are controlled either by step and direction signal or by a quadrature A/B signal. For instance, an automated start up procedure (establishing communication, homing etc.) would still be possible but the actual motion signal generation would need to be integrated by a custom solution developed by the user. To keep the control software as simple as possible it was decided to control the servo motors with software motion control and make use of the complete sFoundation library.

7.2 Control Concept

Initially, position control was considered to be the optimal control concept for the task at hand. Thereby time dependent positions commands are sent to the motor controller which the motors then follow. If the time between two position commands is small enough a smooth motion can be generated. However, the position command available in the sFoundation library executes a trapezoidal move, where the motor ramps up to a defined speed and slows down to a complete stop at the designated position. A second command is only executed if the previous one is completed. As a result, the servo motors would always be at zero velocity after every position command which makes position control impossible for the task at hand. Subsequently velocity control was considered, as the desired motion can be generated by sending time dependent velocity commands to the motor controller. If the acceleration with which each velocity shall be reached is set properly as well, a well-defined motion can be generated. The calculation of the velocity and acceleration values based on the desired time dependent position is done by using simple one-dimensional kinematic equations:

$$\varphi_i = \varphi_{i-1} + \omega_{i-1} * \Delta t + \frac{1}{2}\dot{\omega}_i \Delta t^2 \qquad (7.1)$$

Where φ_{i-1} is the angle at the begin of the step, φ_i is the targeted angle at the end of a step, ω_{i-1} is the angular velocity at the begin of a step and Δt is the duration of a step. φ_{i-1} and φ_i are hereby given by the targeted motion. Rearranging equation x leads to

$$\dot{\omega}_i = 2 * \frac{(\varphi_i - \varphi_{i-1}) - \omega_{i-1} * \Delta t}{\Delta t^2} \qquad (7.2)$$

and

$$\omega_i = \dot{\omega}_i * \Delta t \quad (7.3)$$

Initial tests showed that a maximum of 90 commands per second can be sent to the motor controller with the control software as developed at this point. For the fastest desired motion at 12 Hz this limited the discretisation of the sinusoidal motion to 6 steps per period. The motion commands generated by the control software are buffered in the motion controller and executed either after the previous one is completed or upon a set trigger. As this buffer has 16 slots, an upper limit of 16 steps per period is given. For a sinusoidal motion as needed for the gust generator a discrete motion close to the target can be generated even with only 6 steps per period (see fig. 60). At 16 steps no discrepancy between actual and targeted position is visible anymore (see fig. 61)

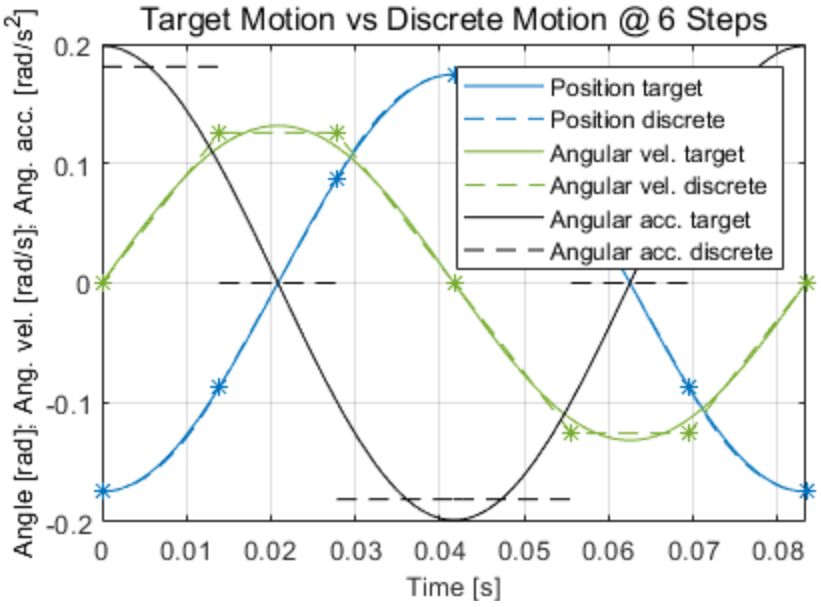


Figure 60: Motion profile at 6 steps per period, acceleration scaling: 1/5000, velocity scaling: 1/100

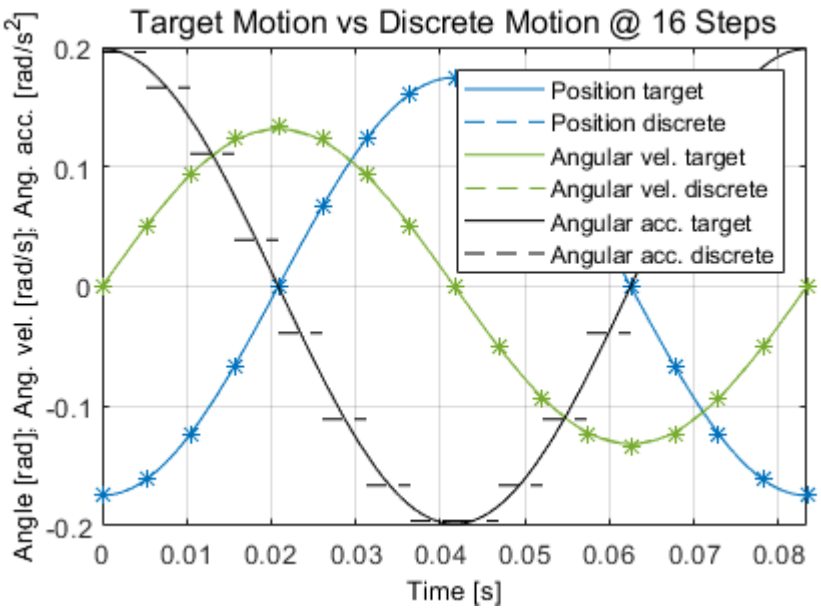


Figure 61: Motion profile at 16 steps per period, acceleration scaling: 1/5000, velocity scaling: 1/100

It must be noted that the velocity control only delivers proper results if the targeted motion is at a velocity of zero at the start of the motion, as jumps in velocity are physically not possible. Using the equation x combined with the start of a motion at a point where de targeted velocity is not zero to begin with leads to oscillation (see fig. 62).

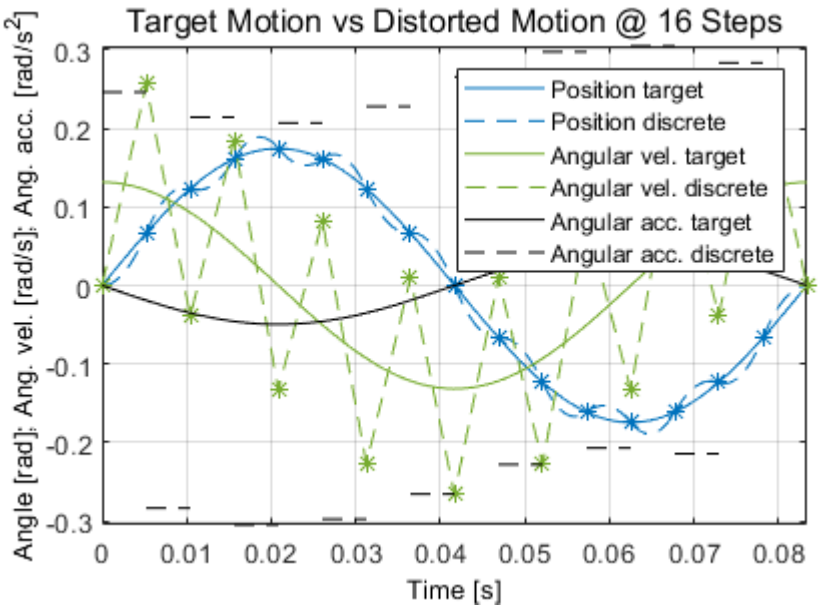


Figure 62: Distorted motion profile due to velocity jump, acceleration scaling: 1/20000, velocity scaling: 1/100

While testing the control concept, it became apparent that the low-level motion-control on the servos need a small increment of time to confirm the target velocity during which the set velocity is held. This led to a small overshoot at every single command leading to the actual motion drifting away from the targeted motion. To address this problems equations, 7.1 and 7.2 were modified assuming that at the end of every velocity command the velocity is held for a short time increment t_l. This led to a correction factor for the velocity as well as for the acceleration applied at each step.

$$\varphi_i = \varphi_{i-1} + \omega_{i-1} * \Delta t + \frac{1}{2}\dot{\omega}_{inew}(\Delta t - t_l)^2 + \dot{\omega}_{inew} * (\Delta t - t_l) * t_l \qquad (7.4)$$

rearranging equation 7.4 leads to

$$\dot{\omega}_{inew} = \frac{\left((\varphi_i - \varphi_{i-1}) - \omega_{i-1} * \Delta t\right) * 2}{(\Delta t - t_l)^2 + (\Delta t - t_l) * t_l * 2} \qquad (7.5)$$

Dividing equation 7.5 by equation 7.2 leads to

$$\frac{\dot{\omega}_{inew}}{\dot{\omega}_i} = \frac{\left((\varphi_i - \varphi_{i-1}) - \omega_{i-1} * \Delta t\right) * 2}{(\Delta t - t_l)^2 + (\Delta t - t_l) * t_l * 2} * \frac{\Delta t^2}{\left((\varphi_i - \varphi_{i-1}) - \omega_{i-1} * \Delta t\right) * 2} \qquad (7.6)$$

simplifying to

$$\frac{\dot{\omega}_{inew}}{\dot{\omega}_i} = \frac{\Delta t^2}{\Delta t^2 - {t_l}^2} \qquad (7.7)$$

The same can be done for ω_i:

$$\omega_{inew} = \dot{\omega}_{inew} * (\Delta t - t_l) \quad (7.8)$$

Dividing equation 7.8 by equation 7.3 leads to

$$\frac{\omega_{inew}}{\omega_i} = \frac{\dot{\omega}_{inew} * (\Delta t - t_l)}{\dot{\omega}_i * \Delta t} \quad (7.9)$$

which can be simplified with by using equation 7.7 to

$$\frac{\omega_{inew}}{\omega_i} = \frac{\Delta t}{\Delta t + t_l} \quad (7.10)$$

Additional to this correction, a position control at every half period was introduced to prevent drift effectively. The complete motion profile is triggered every period to ensure the frequency of the generated motion matches the targeted one. This hybrid approach is shown in figure 63.

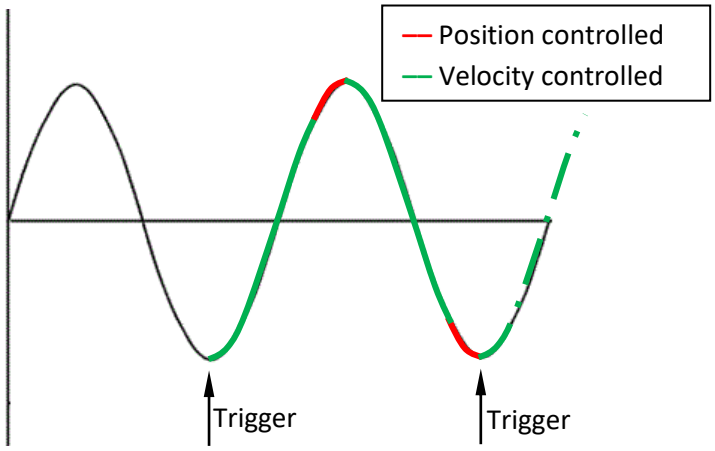


Figure 63: Hybrid motion control

7.3 Software Architecture

The hybrid velocity/position-controlled concept was implemented in the software control. Figure 64 shows a simplified flow chart of the Software architecture.

Upon start-up (1), the user has to enter several parameters to define the motion. Based on these parameters the motion data is generated (2) according to the concept described in chapter 7.2. The sampling rate is hereby defined depending on the frequency as well as on the gust type (3). As the 1-cos gust has a settling period between every gust providing enough time to buffer the next gust completely, any gust can be generated with the highest sampling rate of 16. In a next step communication with the communication hub is established (4). The system will home itself if this is successful (5). The gust vanes are then set to their starting position upon completion of the homing procedure (6). Before the motion can start, an initial start time will be defined based on the current global system time given by the communication hub (7). This has at least 1 ms precision [33]. A second thread is opened executing the trigger function (8). The trigger function generates a simple trigger signal of 3.3 V which can be used to synchronise test equipment with the gust generator. Two trigger mods are available. One allows to trigger always at the same phase angel, the other allows for a phase angle which is changed after a given number of steps. Both threads, trigger and motion, run independently of each other but use the same system time (9) to stay synchronised. The motion thread continuously buffers motion commands onto the motor control unit (MCU) (10) over the communication hub. These motions are repeatedly triggered as described in section 7.2. Both threads are closed as soon as the complete number of cycles is performed. As soon as both threads are finished the software shuts down (11). If at any time a soft limit is hit and a hall sensor switches to high, the motion stops and the control software aborts (12).

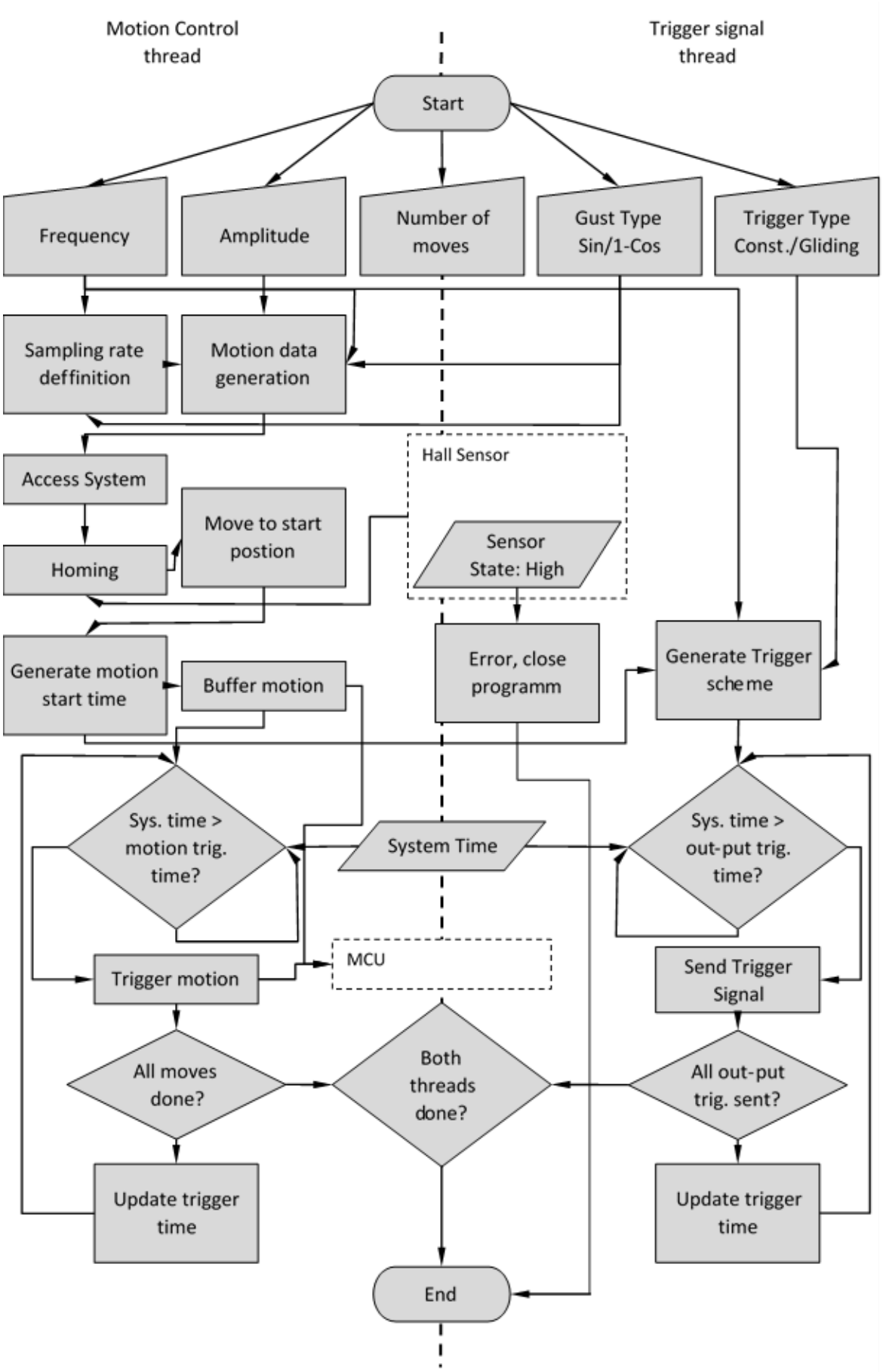


Figure 64: Control software schematic

8 Testing

To validate the CFD data and to characterise the finished gust generator prototype, a test series was performed. The gust generator was attached to the designated wind tunnel, the W-Tunnel at TU Delft. Additionally, the test section was mounted to the gust generator to represent the situation as seen in figure 11 in section 4.2 as well as possible. The flow measurements were done with Particle Image Velocimetry (PIV) to generate a 2D velocity vector field across the test section. The following chapters describe the test setup and procedure.

8.1 Test Method

As mentioned above, the testing was done with PIV, as shown in figure 65 and 66. PIV is based on the detection of particles in the flow and calculates a velocity vector field from the difference between two pictures taken shortly after one another (see fig. 65) [34].

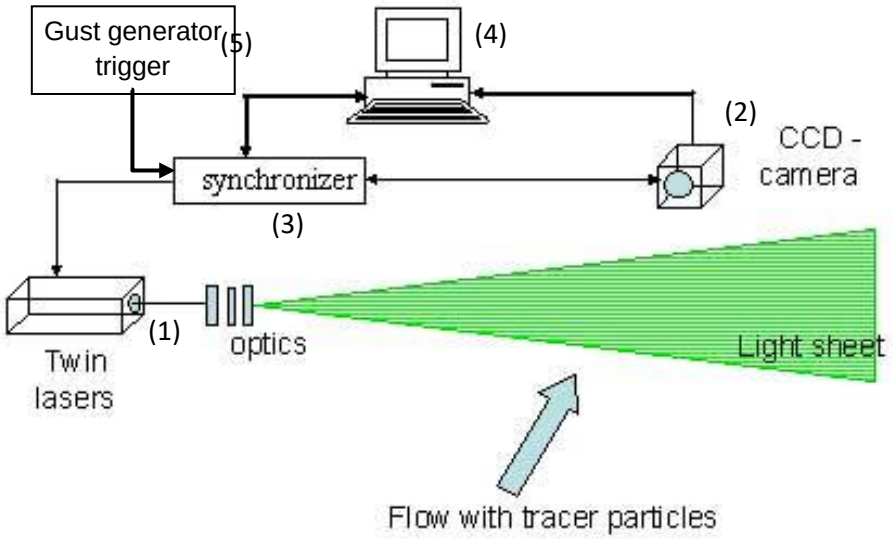


Figure 65: PIV working principle [34]

A commercially available system, apart from the laser provided by LaVision, was used for the testing. The laser (1) used for the tests was a Quantel EverGreen ² 200 provided by Lumibird Group capable of a pulse frequency of 15Hz at a maximum energy of 200 mJ. The pictures (2) were taken with Imager sCMOS cameras capable of a frame rate of up to 50/s and a sensor resolution of 2560 x 2160 pixel. The limiting frame rate was therefore given by the laser with 15 Hz. The synchronisation between camera was done with programable timing unit (3) PTU v9 provided by La Vision as well. The whole system was controlled by Davis 8.4 software running on a computer (4). The phase locking was done with the trigger signal generated by the gust generator control software (5). The actual set-up can be seen in figure (66)

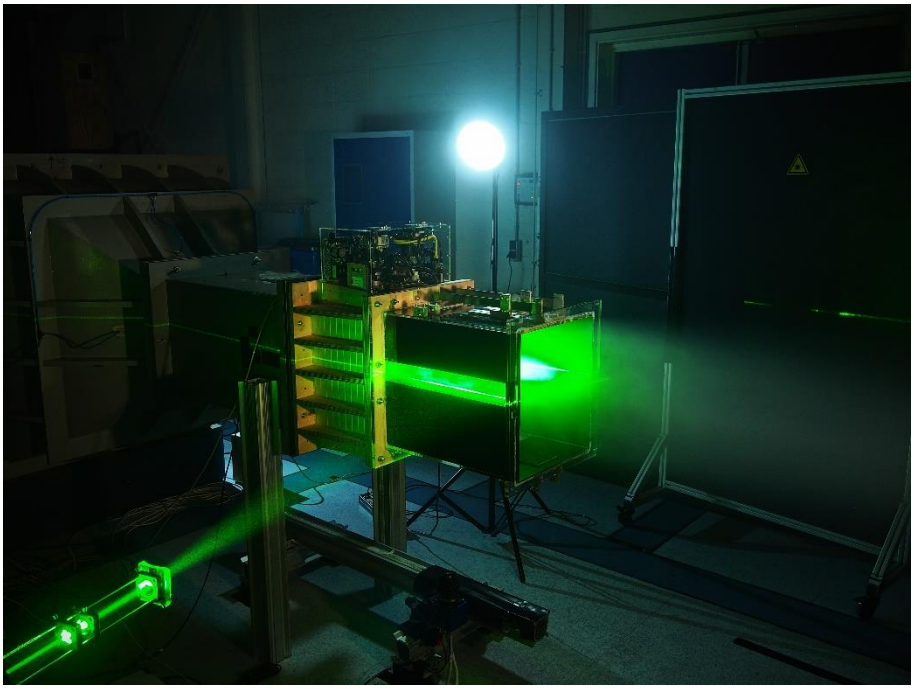


Figure 66: PIV set up as used for experiments

The measurement plane was chosen to be in the x/y- plane as depicted in figure 67. The dashed square herby represents the field of view of the camera of roughly 271.5 x 321 mm. It was aligned as close as possible with the principal axes of the test section and set to be slightly above the half span of the gust vanes.

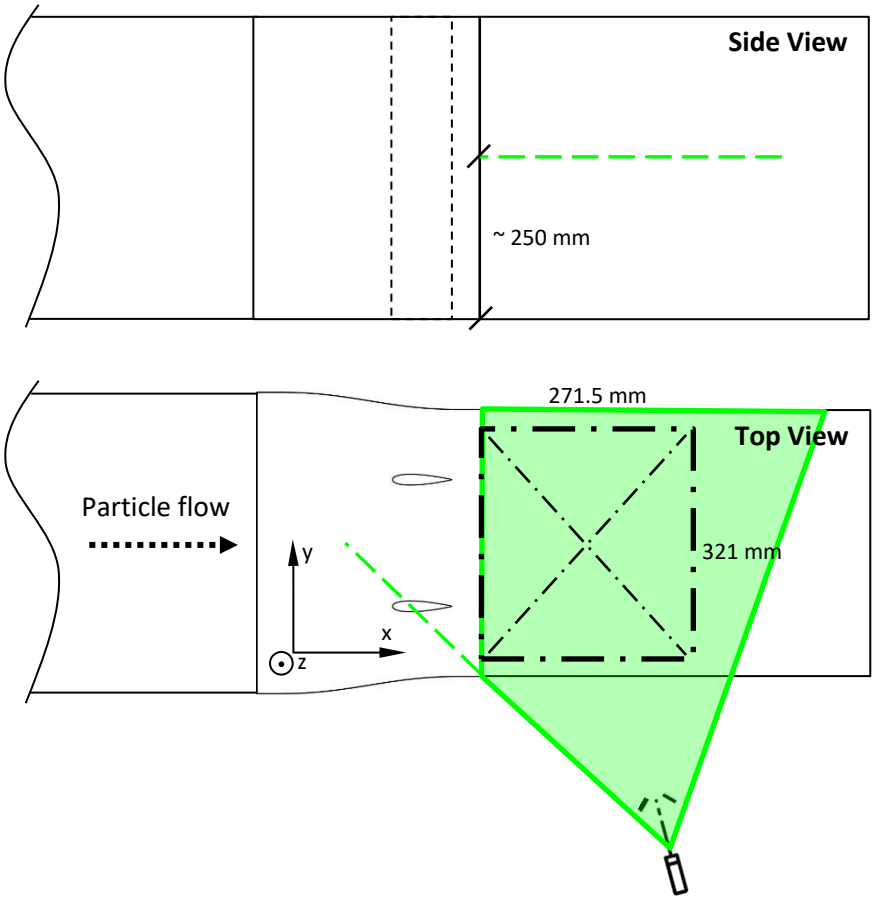


Figure 67: Geometrical overview of PIV Set-Up

8.2 Test Procedure

The test procedure aimed at gathering velocity data for both gust types and over for all reduced frequencies as described in section 5.2.1. Tests were performed at two different air speeds to capture potential phenomenon depending on the flow speed. As mentioned at the beginning of section 8, the goal of the test series was to characterise the gust by generating time dependent gust profiles as well as spatially resolved gust profiles at the time of the maximum gust velocity. Additionally, the test series aimed to validate certain flow phenomenon indicated by the CFD simulations. As the whole set-up has technical limitations with regard to the rate at which data can be gathered, one cannot generate a complete time resolved data set while performing only one gust. Therefore, a series of multiple consecutive gusts was performed for each parameter configuration. The measurement was herby triggered at the same phase angle of the gust vanes for n steps and then repeated at the next phase angle until the complete gust was covered. This process could be performed all in one test series as the control software, if set to moving trigger, continuously updates the trigger position

A total of 12 test series were executed. All tests were performed for 30 m/s air speed as well as for 15 m/s. 1-cos as well as sin gusts were generated and measured. Three different reduced frequencies were investigated. An overview of the performed test series can be seen in table 24.

Flow speed [m/s]	Gust Type	Reduced Frequency (Frequency [Hz])	Step size [deg]	Measurement range [deg]
30	sin	0.2 (12)	18	360
30	sin	0.067 (6)	15	360
30	sin	0.008 (0.5)	15	360
30	1-cos	0.2 (12)	5	390
30	1-cos	0.067 (6)	5	390
30	1-cos	0.008 (0.5)	5	390
15	sin	0.2 (6)	18	360
15	sin	0.067 (2)	18	360
15	sin	0.008 (0.25)	18	360
15	1-cos	0.2 (6)	5	390
15	1-cos	0.067 (2)	5	390
15	1-cos	0.008 (0.25)	5	390

Table 24: Tests overview

As shown in table 24 the sin gusts were each measured over the course of 360° respectively a full motion period, whereas the 1-cos gusts were monitored over 390°. It must be noted that the trigger time is based on the gust vane motion. The gust is however slightly delayed as it needs a certain amount of time until it reaches the point of interest of the measurements. To accommodate for this delay, the measurements were continued for more than one period for the 1-cos gusts. This was not necessary for the sin gusts as they are based on a continuous motion and therefore the resulting measurements are simply slightly phase delayed.

8.3 Post Processing

The raw data generated this way comes in the form of a set of images. These images were then processed with Davis 8.4 to generate the corresponding vector fields as described in section 8.1. This vector fields in form of text files were later postprocessed with multiple MATLAB script in the following manner:

- **Preparation**
  The text files could not directly be processed by MATLAB as commas instead of dots are used as decimal points by Davis 8.4. An initial script only changed every decimal comma to a point.
- **Centreline definition**
  To mitigate any miss alignment of the set-up, measurements done with both gust vane at 0° were used to determine the centre line where y=0:
  1) Load baseline file
  2) Search wake in every data point
  3) Average position of the wakes
  4) Calculate the mean of the wake position
  5) Search for the closest data point with regard to its y – coordinate which is further used as y = 0

  This procedure showed that the system alignment was done very carefully as the y - coordinate of the point defined with this procedure was 0.6503 mm.

- **Phase averaging**
  As described above, data was measured at every phase angle multiple times to allow for phase averaging:
  1) An initial file of the test series needs to be opened to generate the path to the test series.
  2) A new directory is generated to safe the phase averaged data.
  3) The closest data point closest to a defined stream wise position (x -coordinate) is searched. Only data in the stream wise proximity of this point is processed to reduce the amount of data.

  The following procedure is repeated until every file of the series is processed

  4) The data at the defined x - coordinate is collected as well as the data of a defined number of neighbouring data points.
  5) The collected data is averaged. This results in a spatial averaging about the x – coordinate.
  6) This spatial averaged data is collected for all files containing data corresponding to the same phase angle.
  7) All data at the same phase angle is averaged. This results in phase averaged data.
  8) The phase averaged data is written to a new file in the new directory containing the phase angle as information in its file name.
- **Data Collection**
  The phased average data was evaluated with regard of the time resolved velocity data at y = 0 at the stream wise point of interest. Additionally, it was evaluated in a spatial manner at the same stream wise point but only at the time of highest gust amplitude:
  1) An initial file of the phase averaged data is opened to generate the path to the test series.

  The following procedure is repeated until every file of the series is processed

  2) The velocity data at y = 0 is gathered and averaged with a defined number of neighbouring values in y dimension
  3) The velocity data is gathered for every phase angle and saved in a matrix and as a new file

The following procedure is only done once based on the data generated in step 3

4) The absolute maximum gust velocity is searched and the corresponding phase averaged data file is opened again.
5) The complete data of this phase angle is smoothed with a moving average of a defined averaging window.
6) The data is further smoothed with a Savitzky-Golay filter.
7) The Data as of step 5) as well as of step 6) is saved in a new file.

All four MATLAB scripts described above can be found in appendix B.

9 Results

Results regarding mechanical loads, the gust vane motion and the test results obtained by PIV are presented in the following sections.

9.1 Mechanical Load Validation

With the final simulated data of the optimisation and the detailed design available the preliminary calculations could be repeated according to equations 6.1 to 6.14 of section 6.16. With the hinge point being at the leading edge the aerodynamic load is reducing the load on the actuator. Therefore, it was concluded that the worst load case is present when the gust vane angle is 0° where no assisting aerodynamic moment is present, but the dynamic torque is the highest. This is a simplification as the flow is not parallel as established in section 4.3.4. The new values are summarised in table 25. It is apparent that these values are still substantially below of what the hardware can handle.

Parameter	Value
Moment of inertia total assembly [kg*m^2]	$6.32*10^{-4}$
Gearbox Inertia [kg*m^2]	$4*10^{-6}$
Actuator Inertia [kg*m^2]	$0.1*10^{-4}$
Reflected Inertia	$0.1*10^{-4}$
Inertia ratio	1
Torque at gearbox output	0.94 Nm
Torque at actuator	0.15 Nm
Safety factor torque actuator	2
Safety factor torque gearbox	22.3

Table 25: Load validation

9.2 Targeted Gust Vane Motion vs. Performed Gust Vane Motion

As the motion control software is a potential source of errors, the motion generated by the servo motors was logged using the diagnostic port available on them. The motion profile as desired was overlaid over the actual profile as executed by the motors. The results are shown below:

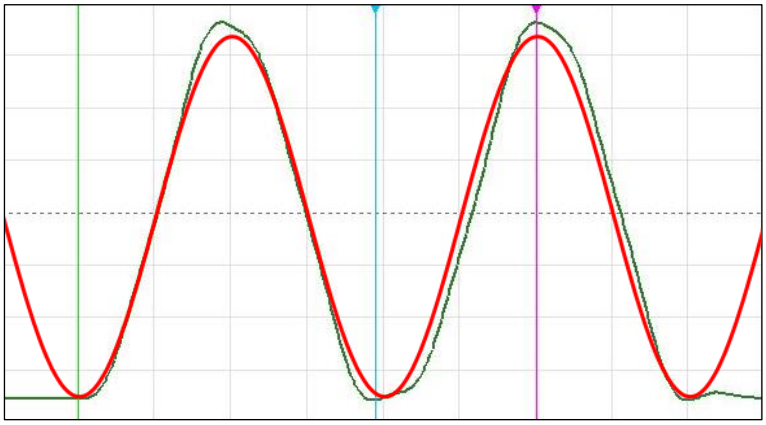


Figure 68: Servo motion, gust type =sin, freq.=12 Hz; red =target, green=executed

Figure 69: Servo motion, gust type =1-cos, freq.=12 Hz; red =target, green=executed

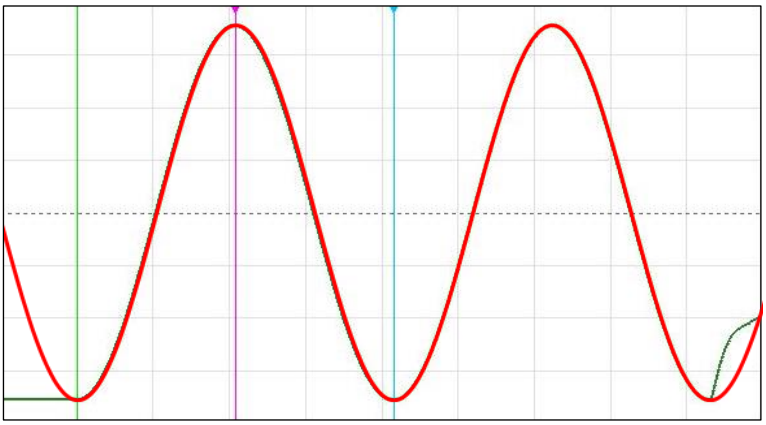


Figure 70: Servo motion, gust type =sin, freq.=0.5 Hz; red =target, green=executed

Figure 71: Servo motion, gust type =1-cos, freq.=0.5 Hz; red =target, green=executed

Overall, a good match between targeted motion and executed motion can be observed. Especially at low frequencies nearly no discrepancy between the targeted motion and the desired motion can be spotted (see fig. 70 and 71), which proves that the chosen controlling concept of a combination of velocity and position control is in principal working well. However, the motion profile at high frequency and continuous motion as performed for a sin gust at 12 Hz shows some discrepancy from the targeted motion (see fig. 68). This is to some extent the result of the low sampling rate of only six motion commands at these frequencies. It also tends to overshoot as the highly dynamic motion leads to higher acceleration and deceleration rates. In contrast, the 1-cos gust motion at 12 Hz as seen in figure 69 follows the targeted motion nearly as well as at low frequencies. This shows that a higher sampling rate of the motion profile improves the result heavily, as all 1-cos gusts are sampled at 16 motion commands per period. Only a slight overshoot as well as some oscillation after the motion can be observed. This could be improved by further fine tuning the low-level motor control. if the sampling rate is high enough. The abrupt change in gust vane angle seen on the right side of figure 70, is not the result of a control error but is simply the un-loaded gust vane moving into the flow after the motion has been completed.

9.3 PIV Results

Only a selection of the complete data set is shown to highlight the most important results. The raw data available would allow for a more in-depth analysis of the gusts as generated by the prototype but this would go beyond the scope of this thesis. Figure 72 shows a gust at extremely low flow speeds of 2 m/s with a correspondingly short wavelength. The displayed data reflect v_y. Figure 73 shows an unprocessed Image of the same run with two shed vortices clearly visible.

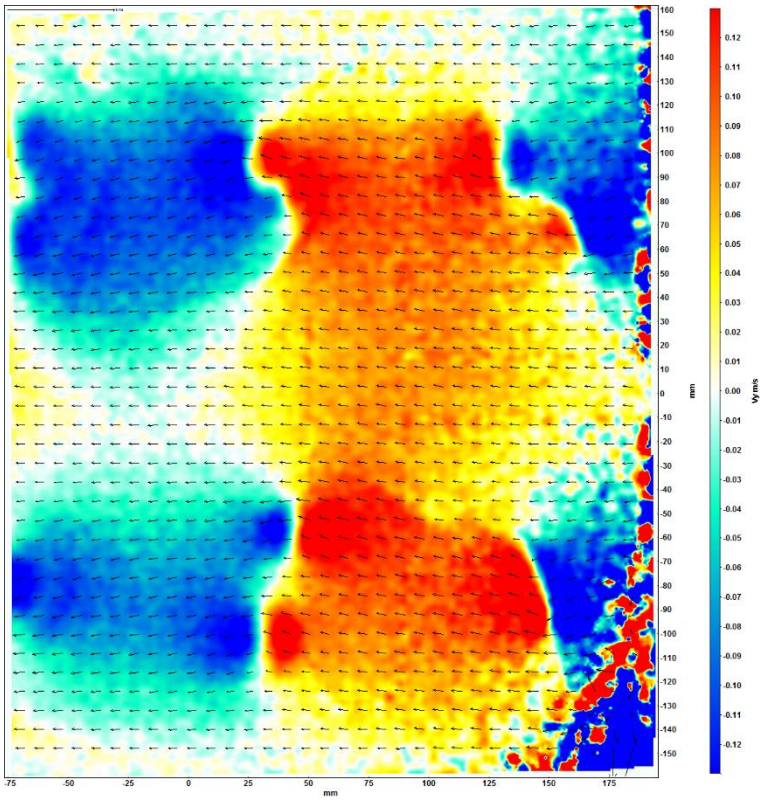


Figure 72: Low speed gusts, vector field

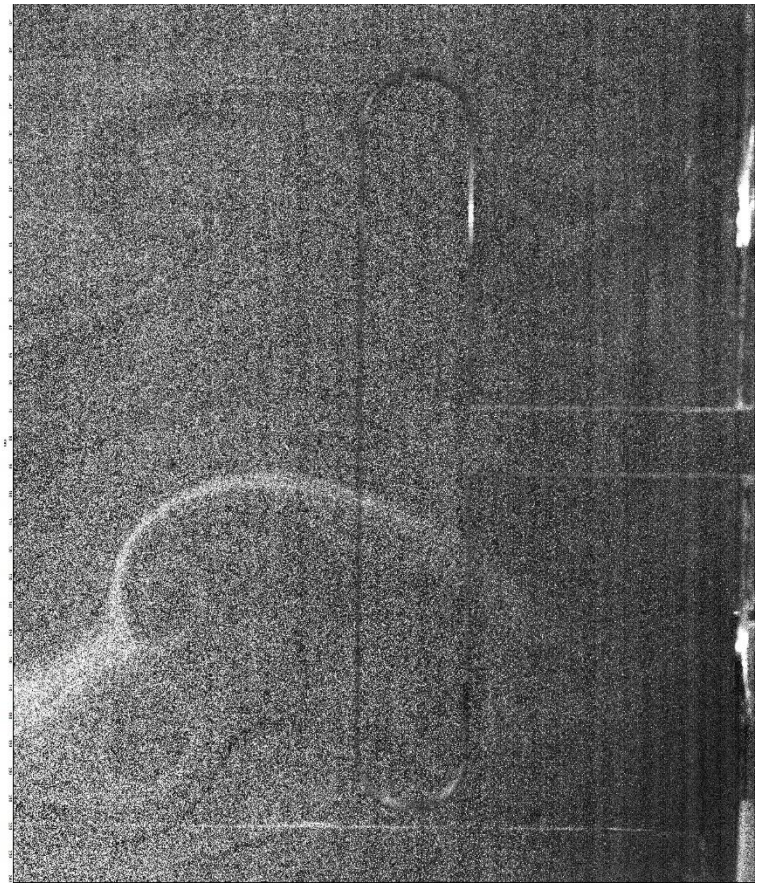


Figure 73: Particle flow, low speed gust

9.3.1 Gust Characteristics

Using the velocity data generated during the test series multiple flow parameters were calculated. Table 26 shows a summary of these parameter.

Data Source	Flow speed [m/s]	Max. gust velocity [m/s] unsmoothed/smoothed	Max. gust deviation [%] unsmoothed/smoothed	Wavelength [m] 12 Hz/4 Hz/0.5 Hz 6 Hz/2 Hz/0.25 Hz
Sim. at 30 m/s	34.3	1.476/-	3.42/-	2.86/5.71/68.6
PIV at 30 m/s	34.1	1.83/1.89	9/8.7	2.84/5.68/68.2
PIV at 15 m/s	17.1	-/-	8.2/8.4	2.85/5.7/68.4

Table 26: Numerical results of PIV Testing

It is obvious that the flow speed which is measured in the test section is roughly 14 % higher than the set air speed. This is a direct consequence of the narrowing shape of the gust generator itself. Simulation and measurements are in agreement with regard to this property. The maximum measured gust velocity is higher than the predicted by CFD. These results are however difficult to be compared in quantitative manner, as the PIV measurements could not be evaluated exactly at the same location as the CFD simulation and the gust strength is highly dependent on stream-wise direction as described in section 5.2.1. The measured data could not be evaluated at the exact same location as a cut out in the test section generated some reflection it this area which subsequently tampered the data quality.

Qualitatively speaking one can state that the gust velocity most likely is higher in the gust generator then anticipated based on the simulations, as the point at which the CFD data was logged and the point at which the PIV data was processed are in proximity of each other. The measured gust deviation appears to be substantially bigger as the expected one with respect to the simulation. This could be based to some extent on noisy data and therefore must at this point be seen as an initial assessment. Further explanation can be found in the next section.

9.3.2 Simulated Gust Profile vs. Tested Gust Profile

To further asses the characteristics of the gust generator as well as how it compares to the simulation results, the gust velocity was plotted over time as well as along the y – dimension at the time step of highest absolute gust velocity. Note that the gust profiles were normalised and orientated to match the configuration of the CFD simulation (see fig. 74 and 75). The time resolved data was herby normalised with the negative maximum of the gust velocity, whereas the spatial resolved data was normalised with the gust velocity at y = 0.

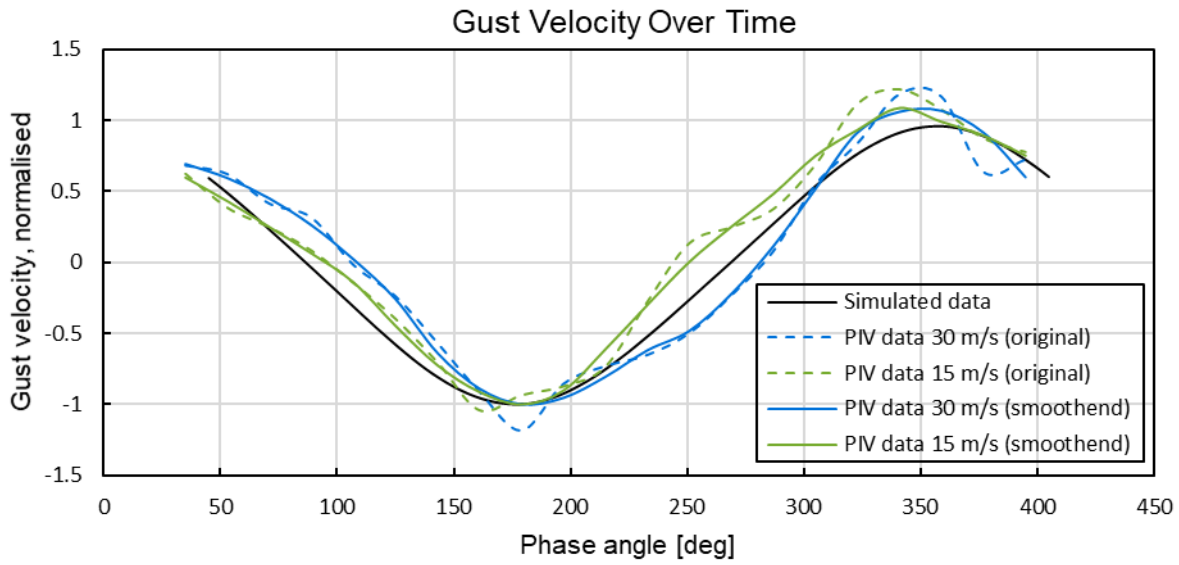


Figure 74:Gust velocity over time for a reduced frequency of 0.2

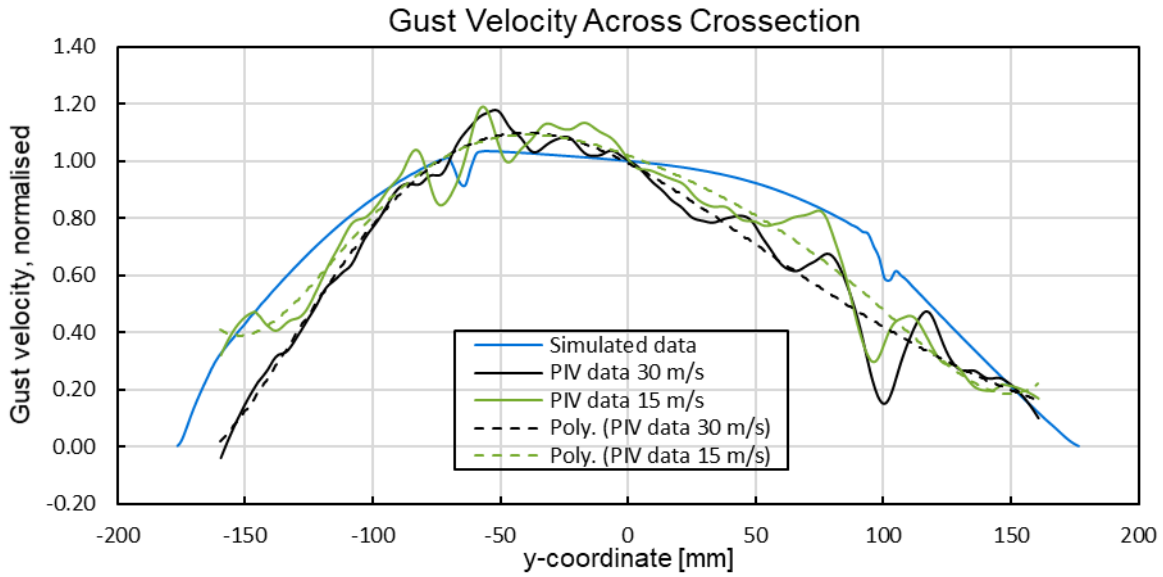


Figure 75: Gust velocity over y-coordinate at maximum gust angle for a reduced frequency of 0.2

As mentioned earlier, the obtained data allowed only for an initial qualitative assessment the gust quality. A more precise quantitative assessment of the gust quality would require a more in-depth analysis of the data as well as additional testing. The time resolved as well as the spatially resolved gust profile as measured using PIV are in the range of what was expected based on the CFD simulations and are in general of comparable shape. The time resolved data does not differ dramatically from the simulated data, however a closer match was expected. In contrast, the spatial resolved data is a lot noisier and the mismatch is more pronounce. A few possible reasons for the discrepancy between simulated and measured data could be identified.

The discrepancy of the time resolved data could be based on the triggering of the PIV system. If the trigger is imprecise, irregular delays could occur and thus lead to measurements not done at the targeted exact phase angle. If this delay is not constant, phase averaged results will be subjected to a systematic random error. At high motion frequencies moderate vibrations were observed, which most likely had an impact on the time resolved results. These vibrations were not anticipated and were based on the complete set-up as the gust generator together with the test section form a significant overhang (see fig. 3). This overhang was supported in vertical but not in horizontal direction, thus allowing for vibration in the x/y – plane.

As the data could only be post processed for a limited amount of time in the framework of this thesis, the quality of the data could be improved for both the spatial as well as the temporal resolved gust profiles with a more refined post processing routine. As figure 75 clearly shows, the spatial resolved data is affected by noise. This noise can only be partially based on a potential unprecise trigger signal. It is far more likely that its main cause is the to some extent over resolved turbulent flow as seen in figure 76. The negative effect of these fluctuations on the final result could be minimized with the suggested more refined postprocessing routine. Hence, as mentioned earlier, the results are of preliminary nature and are predominantly meant to be used for a qualitatively assessment.

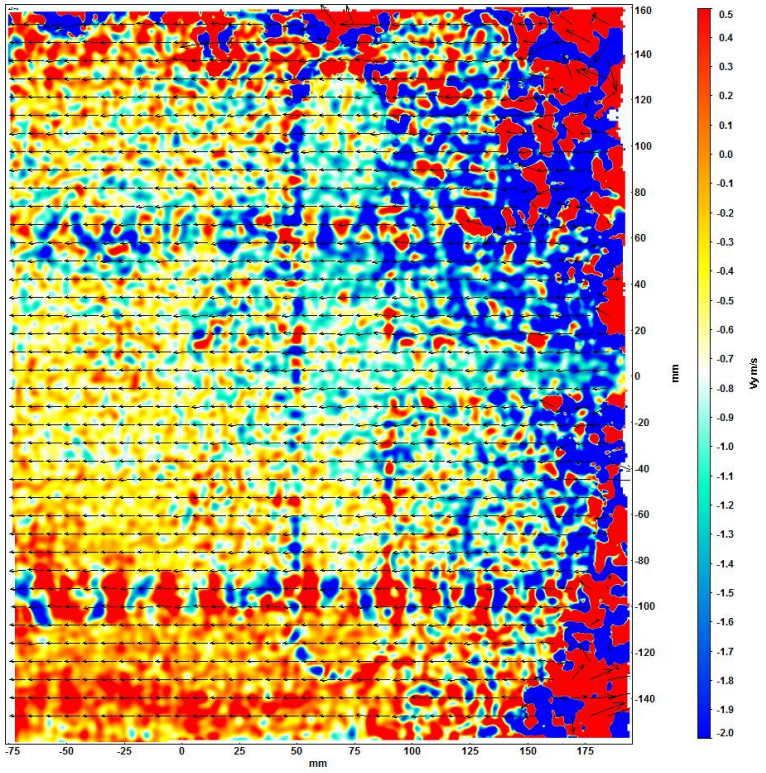


Figure 76: PIV results, vy dependent colouring

9.3.3 Frequency Related Inversion of the Gust Velocity due to Vortex Shedding

The phenomenon of a gust velocity inverse to the intuitively expected direction at the beginning and the end of a gust vane motion was investigated with the data of 1-cos gusts. As figure 77 shows, this behaviour could be reproduced. It must be noted that the phenomenon could only be captured at the end of the gust. It seems as if the trigger is delayed to a certain extent and subsequently prevented the capture of the whole complete gust profile. It is clearly visible that this behaviour is highly correlating with the reduced frequency as already described in section 5.1. It must be noted that after every gust motion the gust vanes were kept steady for a certain time allowing the flow to settle. Thus, the shown behaviour can in fact be interpreted as the inversion of the gust velocity and is not just the start of a consecutive gust.

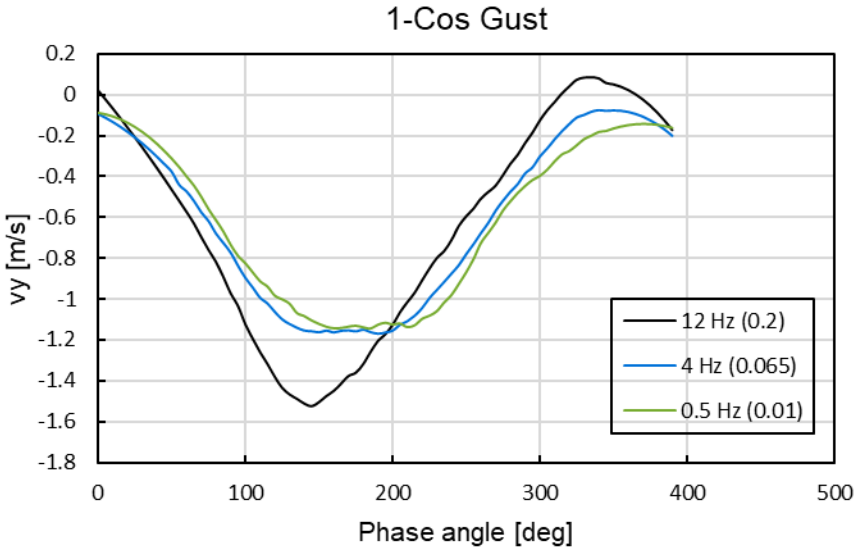


Figure 77: Reduced frequency dependent revers flow at 1-cos gust

9.3.4 Gust Angle dependency on Reduced Frequency

The last aspect which was addressed was the independency of the gust angle of the flow speed. It is stated in literature [4] that the gust angle is only dependent on the reduced frequency but not on the flow velocity or the frequency itself. Smoothened data was used to mitigate distortion due to noise. As figure 78 shows this could be shown nicely for the two higher reduced frequencies. The measurements at the lowest reduced frequency do however not support this statement. A closer look at the time resolved data did however show unintended flow behaviour. When the gust is approaching its maximum it suddenly drops back to lower levels (see fig. 79).

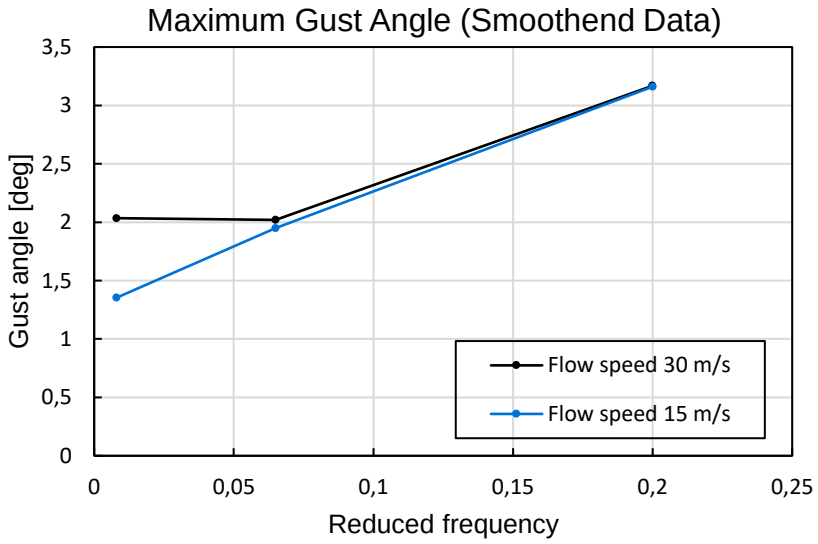


Figure 78: Gust angle vs reduced frequency

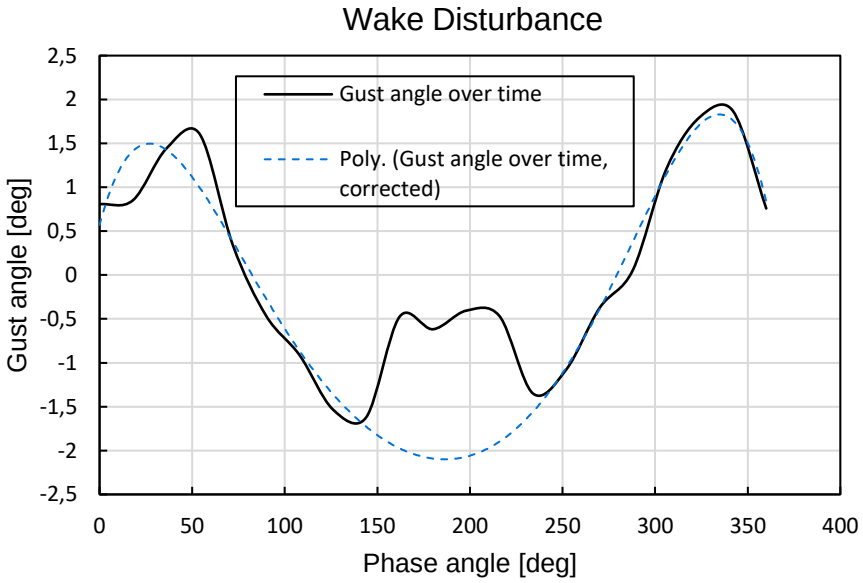


Figure 79: Gust decrease due to wake turbulence

This behaviour is based on the flow separating of the airfoil at low speed and in quasi steady state, creating a heavy wake which distorts the measurements. (see fig. 80). Flow separation always happens first at the gust vane with the trailing edge pointing towards the wall as this gust vane experiences a higher angle of attack due to the narrowing geometry of the gust generator. A curve was fitted to the data to estimate the gust angle if no separation would occur. This virtual gust angle would again be in agreement with the initial statement.

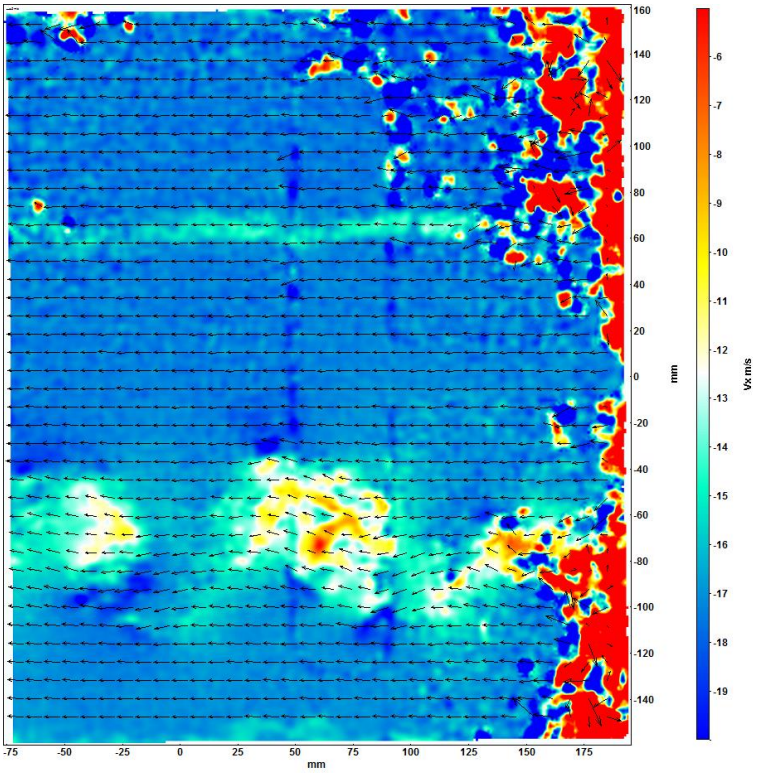


Figure 80: Wake at low flow speeds and maximal gust vane angle

10 Conclusion

The goal of the present work was to develop a gust generator capable of producing sin and 1-cos gusts. The gust generator should be usable in combination with the W-Tunnel at TU Delft Faculty of Aerospace Engineering. The system should be fully enclosed and be usable solely with a computer connected either physically or over a network. These goals could be reached and a final working prototype was developed, built and tested. A more detailed conclusion for multiple aspects of the thesis is given in the following sections.

10.1 Design

The mechanical design focused on a system of high rigidity combined with as little free play as possible. Additionally, it should be possible to disassemble the gust vanes to allow for multiple configurations. The final mechanical design fulfilled all these goals. The chosen hardware components are able to perform the highly dynamic motions with ease and still show a large reserve with regard to torque as well as maximum speed and acceleration. This allows for high flexibility in the future development of more demanding motion profiles.

10.2 Control Software

The hybrid velocity and position control for the motion proved to be working and to allow for a motion profile close to the targeted profile down to as little as 6 motion commands per period. The concept could be implemented in a control software which allows to generate sin as well as 1-cos gusts. However, it became apparent that the software performance at high frequencies needs improvement. The rate at which motion commands can be forwarded to the servo motors appears to be lower than what the manufacturer promises, indicating the possibility of bottlenecks in the software. Additionally, the software has showed some stability issues making testing at high frequencies somewhat difficult. These issues are related to timing and the management of the motion buffer used by the low-level motor control. The incorporated trigger seemed to work as intended on initial trial, but the gathered PIV data indicates that the timing is limited in accuracy. Further work is thus necessary. Summarised one can say that the control software in its current state enables the gust generator to be used as intended but multiple aspects of it need further development to increase its performance.

10.3 Optimisation

The extended CFD study which was performed as part of the design optimisation predicted plausible results and multiple measures were taken to ensure reliable results. Multiple phenomena described in literature could be reproduced. Somewhat unique flow characteristics due to the narrowing cross section of the gust generator can be assumed due to the simulation results. It seems that the proximity of the gust vanes to the walls can be used to compensate some draw backs due to this geometrical circumstance, leading to a gust of high uniformity around the centre line between the gust vanes. An optimisation with regard to the positioning of the gust vanes could be performed and resulted in a final placement of the gust vane at a position where a gust of high uniformity was indicated by the CFD simulation. However, a final validation of the transient simulation results is still to be done as the test results based on PIV data is inconclusive as of now.

10.4 Test Results

Clean and precisely installed test equipment allowed for the generation of PIV raw data of high quality. Qualitative statements with regard to the gust characteristics could be made. Additionally, certain flow phenomenon could be observed. The inverted gust velocity at the beginning and the end of 1-cos gust at intermediate to high reduced frequencies as seen in the simulation could be reproduced. The correlation between gust angle and reduced frequency could be shown as well. The test results also indicate certain lower limits for frequencies and flow speeds as conditions can be generated where the flow fully separates from the airfoil and subsequently a big wake is formed which heavily affects the gust.

Multiple circumstances could be identified which in the end led to quantitatively inconclusive test results. Vibration at high frequencies and to some extent difficulties with the gust generator control software, especially with the trigger signal generation, let to data points which must be assumed to not be precisely synchronised with the gust vane motion, subsequently affecting the phase averaging and finally the time resolved gust profile. The applied post processing routine is not tuned enough to compensate such flaws in the collected data. The spatial averaging as part of the postprocessing was not able to sufficiently smoothen out turbulence in the flow leading to noisy spatial resolved gust profiles. Therefore, further test series with improvements in the set-up as well as a refined postprocessing routine seem to be necessary to generate more conclusive quantitative results.

11 Outlook

The present work focused on the development of a gust generator capable of performing sin and 1-cos gusts. In that sense the final system must be seen as a prototype. Improving certain aspects of the system is therefore encouraged. Furthermore, the specification of the system allows for additional functionalities with regard to the generation of gusts. The following sections provide an insight which aspects of the system could be improved or extended and how this could be done.

11.1 System Simulation

The system was so far only simulated as part of a 2D CFD simulation. As the system itself was considered to be rigid enough, no FSI simulation was performed. The final tests showed that this assumption was right, however the whole assembly of wind tunnel/gust generator/test section showed vibration at high motion frequencies. An FSI simulation or at least a dynamic mechanical simulation should be performed to get a better understanding of these vibration. Appropriate measures to prevent these vibrations can then be taken.

The optimisation procedure performed as part of this development project did only consider the gust generated inside the empty test section, as it was considered reasonable to treat the generated gust independent of any test case. This was done as such to prevent any unwanted bias of the experiment by tailoring the gust and with it the gust generator to the experiment itself. Nevertheless, it occurred that certain boundary conditions for the gust generator are directly influenced by the experimental set-up. For any future development projects regarding a gust generator it is therefore advisable to include an aeroelastic characterisation of the complete system by theoretical means and/or by performing FSI simulations. Depending on the mechanical properties it can be acceptable to assume the gust generator components them self as rigid to simplify the model. It must be clearly defined what results shall be derived from such simulations and how they will influence a potential design optimisation process.

11.2 System Characterisation

So far only qualitative test results could be derived. A further in-depth analysis of the available raw data is therefore advised for two reasons. With an improved post processing routine one can generate more conclusive results. Additionally, such an analysis could help with identifying any systematic error in the measurements which then can be addressed. A systematic approach is necessary to identify the source of the erroneous data. It is suggested to start by monitoring the trigger signal as generated by the gust generator and comparing it to the gust vane motion to assess if the trigger signal is emitted at the correct phase angles. In a second step it is advised to take measures to prevent the horizontal vibration of the gust generator and the attached test section. For conclusive characterisation as well as for a proper validation of the transient CFD simulations, additional test runs will be necessary after potential problems with the set-up have been resolved.

11.3 Control Software

The control software is not running as stable as desirable. Therefore, it should be thoroughly checked to identify any bottleneck or timing issue. If stability problems persist it is advised to consider a switch from the current software-controlled motion to a step and direction controlled motion. This is possible with the currently installed hardware.

11.4 Future System Upgrades

As the current system was developed with focus on flexibility for further improvements, multiple possible upgrades could be realised. Overlaying an opening and closing motions of both gust vanes with a regular gust motion could result in a gust in two direction. The velocity would not only change in y -direction but also in x – direction. Inside the mechanical limits countless options are possible.

Bibliography

[1] F. M. Hoblit, *Gust Loads on Aircraft: Concepts and Applications*. Washington, D.C, USA: American Institute of Aeronautics and Astronautics, Inc., 1988.

[2] 'Certification Specifications for Normal, Utility, Aerobatic, and Commuter Category Aeroplanes CS23 Amendment 3'. European Aviation Safety Agency, Köln, Germany, 2012.

[3] Z. Wu, Y. Cao, and M. Ismail, 'Gust loads on aircraft', *Aeronautical Journal*, vol. 123, no. 1266, pp. 1216–1274, 2019.

[4] P. M. G. J. Lancelot, J. Sodja, N. P. M. Werter, and R. De Breuker, 'Design and testing of a low subsonic wind tunnel gust generator', *Advances in Aircraft and Spacecraft Science*, vol. 4, no. 2, pp. 125–144, 2017.

[5] P. Lancelot, J. Sodja, and R. De Breuker, 'Investigation of the unsteady flow over a wing under gust excitation', 2017.

[6] P. G. A. Cizmaş, D. Tang, and E. H. Dowell, 'Flow about a slotted cylinder-airfoil combination in a wind tunnel', *Journal of Aircraft*, vol. 33, no. 4, pp. 716–721, 1996.

[7] H. Xu, S. Xing, and Z. Ye, 'Numerical study of an airfoil/rotating-slotted-cylinder based flutter exciter', *Journal of Aircraft*, vol. 52, no. 6, pp. 2100–2105, 2015.

[8] J. M. Roadman and K. Mohseni, 'Gust characterization and generation for wind tunnel testing of micro aerial vehicles', 2009.

[9] R. E. G. Poorte and A. Biesheuvel, 'Experiments on the motion of gas bubbles in turbulence generated by an active grid', *Journal of Fluid Mechanics*, vol. 461, pp. 127–154, 2002.

[10] J. Sodja, F. Roizner, R. De Breuker, and M. Karpel, 'Experimental characterisation of flutter and divergence of 2D wing section with stabilised response', *Aerospace Science and Technology*, vol. 78, pp. 542–552, 2018.

[11] R. E. Sheldahl and P. C. Klimas, 'Aerodynamic characteristics of seven symmetrical airfoil sections through 180-degree angle of attack for use in aerodynamic analysis of vertical axis wind turbines.', Albuquerque, USA, 1981.

[12] R. Acharya, 'Investigation of Differences in Ansys Solvers CFX and Fluent', Royal Institute of Technology, KTH, 2016.

[13] 'Fluent User's Guide'. Ansys Inc., Canonsburg, USA, 2020.

[14] 'CFX'. Ansys Inc., Canonsburg, USA, 2019.

[15] H. R. Schneebeli and A. Vogelsanger, 'Die ICAO-Atmosphäre als Datenmodell'. SwissEduc, Wettingen, Switzerland, 2017.

[16] P. Jenny, 'Turbulent Flows'. ETH Zürich, Zürich, Switzerland, 2017.

[17] Dr. Aidan Wimshurst, 'Fluid Mechanics 101', 2019. https://www.fluidmechanics101.com/.

[18] Dr. Aidan Wimshurst, 'Fluid Mechanics 101 Calculators & Tools'. Fluid Mechanics 101, 2019.

[19] A. Kapoor, 'A Quick Approach to Boundary Layer meshing', 2016. https://www.linkedin.com/pulse/quick-approach-boundary-layer-meshing-beginners-awadh-

kapoor-.

[20] ‘Meshing User’s Guide’. Ansys Inc., Canonsburg, USA, 2019.

[21] ‘Fluent 12.1 User’s Guide’. Ansys Inc., Canonsburg, USA, 2009.

[22] D. Lindblad, ‘Implementation and run-time mesh refinement for the k - omega SST DES turbulence model when applied to airfoils’. Chalmers University of Technology, Göteborg, Sweden, 2014.

[23] D. C. Eleni, T. I. Athanasios, and M. P. Dionissios, ‘Evaluation of the turbulence models for the simulation of the flow over a National Advisory Committee for Aeronautics (NACA) 0012 airfoil’, *Journal of Mechanical Engineering Research*, vol. 4, no. 3, pp. 100–111, 2012.

[24] T. Adams, C. Grant, and H. Watson, ‘A Simple Algorithm to Relate Measured Surface Roughness to Equivalent Sand-grain Roughness’, *International Journal of Mechanical Engineering and Mechatronics*, vol. 1, no. 1, pp. 66–71, 2012.

[25] N. Pugno, E. Lepore, S. Brianza, F. Antoniolli, M. Buono, and A. Carpinteri, ‘Preliminary in vivo experiments on adhesion of geckos’, *Journal of Nanomaterials*, vol. 2008, no. 1, 2008.

[26] S. Hiziroglu and S. Suzuki, ‘Evaluation of surface roughness of commercially manufactured particleboard and medium density fiberboard in Japan’, *Journal of Materials Processing Technology*, vol. 184, no. 1–3, pp. 436–440, 2007.

[27] P. Romanski and M. Burdek, ‘The influence of extrusion speed on selected surface roughness parameters of aluminium profiles’, Częstochowa, Poland; Gliwice, Poland, 2014.

[28] M. Drela and H. Youngren, ‘XFoil 6.9’. MIT, Cambridge, USA, 2001.

[29] M. Kornhaas, D. C. Sternel, and M. Schäfer, ‘Influence of time step size and convergence criteria on large eddy simulations with implicit time discretization’, in *Quality and Reliability of Large-Eddy Simulations*, vol. 12, Luxemburg, Luxemburg: Springer Science+Business Media B.V., 2008, pp. 119–130.

[30] A. V. Kassem, ‘Wind Gust Generation for Wind Turbine Testing via Numerical Methods Ad Wind Gust Generation for Wind Turbine Testing via Numerical Methods’, TU Delft, 2019.

[31] K. Knight, ‘Understanding Inertia and Reflected Inertia’. MCMA, Ann Arbor, USA, 2015.

[32] H. Gill, ‘Stepper Motor or Servo Motor : Which should it be?’ Kollmorgen, Köln, Germany, 2016.

[33] ‘s-Foundation Library’. Teknic Inc, Victor, USA, 2017.

[34] Velocimetry, ‘Particle Image Velocimetry’, 2008. https://www.velocimetry.net/principle.htm.

Appendix A

Table 27: XFoil data for NACA 0018

Angle of Attack [deg]	Cl	Cd
-20	-0.7011	0.22281
-19	-0.6725	0.21018
-18	-0.6444	0.19741
-17	-0.6198	0.18396
-16	-1.2136	0.05538
-15	-1.2186	0.04621
-14	-1.1862	0.04003
-13	-1.136	0.03508
-12	-1.0924	0.03088
-11	-1.0526	0.02737
-10	-1.0256	0.02488
-9	-0.9945	0.02272
-8	-0.9572	0.02097
-7	-0.9084	0.01957
-6	-0.8163	0.01855
-5	-0.6531	0.01763
-4	-0.497	0.01661
-3	-0.3465	0.01549
-2	-0.2192	0.01442
-1	-0.1069	0.01369
0	0	0.01344
1	0.107	0.01369
2	0.2192	0.01442
3	0.3465	0.01549
4	0.497	0.01661
5	0.6531	0.01762
6	0.8164	0.01854
7	0.9082	0.01956
8	0.9571	0.02096
9	0.9944	0.02272
10	1.0257	0.02488
11	1.0531	0.02737
12	1.0931	0.03087
13	1.1371	0.03508
14	1.1887	0.04002
15	1.2196	0.04623
16	1.2165	0.05528
17	1.1841	0.06913
18	1.1672	0.0831
19	0.6746	0.21067
20	0.7033	0.22343

Appendix B

CFD_Postprocess_Optimisation

```
%% Post Processing Routine for averaged DIC Strain Data

clc
close all
clear all
%% General Input

SmplNoD1=4; %Number of samples in X Direction
SmplNoD2=4; %Number of Samples in Y Direction
SampleFirst=1; %Number of First Sample to Process
SampleLast=16; %Number of Last Sample to Process (equal SampleFirst if only
one should be processed)
WofWake=0.022; %width of Wake (Guessd of X-Velocity Graphs)
CellNoY=176; %No of data points on Y axis
DeltaYMin=80; %Min Distance between Gustvanes (optimization points) in mm
DeltaDeltaY=64; %Step size for DeltaY between different Samples
XMax=300; %Max distance in flow direction from GG exit (optimization points)
in mm
DeltaX=72; %Step size for X between different Samples
WToCMin=0.03; %Min distance of wake to center required to ensure that TW
stays in gust zone
TToEndMin=0.04; %min distance of GV Wing end to end of Gust Generator based
on Turbulence intesity behind GV
IPNo=7; %Number of additional Data Points inbetween two existing points (in x
and y direction)
SmplNoD1New=SmplNoD1+(SmplNoD1-1)*IPNo; %New Number of samples in X direction
after interpolation
SmplNoD2New=SmplNoD2+(SmplNoD2-1)*IPNo; %%New Number of samples in Y
direction after interpolation
GVc=0.08; %Gust vane chordlength
WVGAmax=0.2; %Weigth for MaxVelocity/GustAngle
WVGADev=0.8; %Weigth for Velocity/GustAngle Deviation



%% Generate Empty Matrixes

% Matrix Vx
VxMin = zeros(CellNoY,SmplNoD1*SmplNoD2);
% Matrix Vx
VxMax = zeros(CellNoY,SmplNoD1*SmplNoD2);

% Matrix Vy Min
VyMin = zeros(CellNoY,SmplNoD1*SmplNoD2);
% Matrix Vy Max
VyMax = zeros(CellNoY,SmplNoD1*SmplNoD2);

% Matrix Min Gust Angle
GAMin = zeros(CellNoY,SmplNoD1*SmplNoD2);
% Matrix Max Gust Angle
GAMax = zeros(CellNoY,SmplNoD1*SmplNoD2);

% Matrix of Y Coordinates
YCoord = zeros(CellNoY,SmplNoD1*SmplNoD2);
```

```
% Matrix Vx(Single Data)
VxMinSingle = zeros(CellNoY,1);
% Matrix Vx, Max Gust Angle (Single Data)
VxMaxSingle = zeros(CellNoY,1);

% Matrix Min Vy(Single Data)
VyMinSingle = zeros(CellNoY,1);
% Matrix Min Vy(Single Data)
VyMaxSingle = zeros(CellNoY,1);

% Matrix Max Gust Angle(Single Data)
GAMinSingle = zeros(CellNoY,1);
% Matrix Max Gust Angle(Single Data)
GAMaxSingle = zeros(CellNoY,1);

% Matrix of Y Coordinates (Single)
YCoordSingle = zeros(CellNoY,1);

%% Iterative Data Collection

[fileOverTime,pathOverTime]=uigetfile('*.out','Chose any "0.07 timeresolved
file" inside the folder structure over which you want to optimize:');%chose
initial File in any folder, Folder and file names need to suited for this
code

for c=SampleFirst:SampleLast
    %% Data Collection

    % Acces data files (of all data sets)
    strcounter=append(num2str(c),'\');

    if c<10
        pathOverTime=pathOverTime(1:end-2); % Generate path to different sim
results folders
        pathOverTime=append(pathOverTime,strcounter);
    else
        pathOverTime=pathOverTime(1:end-3);
        pathOverTime=append(pathOverTime,strcounter);
    end

    fileOverTime(19)='*'; % generate file name independent of opt1 or opt2
for "Vy over time @ hinge point of test wing"
    fileOverTimeArray=dir([pathOverTime fileOverTime]);
    fileOverTimeArray = {fileOverTimeArray.name};
    fileOverTime=fileOverTimeArray{1,1};

    OverTime=readmatrix([pathOverTime fileOverTime],'FileType','text',
'Delimiter',' ','OutputType','double'); % print data to matrix

    % Search for simulation time of min and max gust angle
    [pksMax,locsMax] =
findpeaks(OverTime(:,2),OverTime(:,1),'MinPeakDistance',10,'MinPeakProminence
',0.01); %find points of max gust angles
    [pksMin,locsMin] = findpeaks(-
OverTime(:,2),OverTime(:,1),'MinPeakDistance',10,'MinPeakProminence',0.01);%f
ind points of min gust angles
    pksMin = pksMin*(-1);

    locsMaxLast=locsMax(end); %only take last min and max (where gust
setteld)
```

```
    pksMaxLast=pksMax(end);
    locsMinLast=locsMin(end);
    pksMinLast=pksMin(end);

%%Sanity Check
%{
     plot(locsMinLast,pksMinLast,'o')
     hold on
     plot(locsMaxLast,pksMaxLast,'o')
     plot(OverTime(:,1),OverTime(:,2))
%}

    % Acces data of Vy over y corresponding to above timestep
    fileOverY = 'xy_velocity_tw_topt*-*'; %generate dummy file name
independent of opt1 or opt2 for gust over y according to max and min
    fileOverYArray = dir([pathOverTime fileOverY]);
    fileOverYArray = {fileOverYArray.name};
    fileOverYMax=fileOverYArray{1,locsMaxLast}; %generate filename of file
corresponding to above timesteps
    fileOverYMin=fileOverYArray{1,locsMinLast};

    OverYMax=readmatrix([pathOverTime fileOverYMax],'FileType','text',
'Delimiter',',','OutputType','double'); %Write data to matrix
    OverYMin=readmatrix([pathOverTime fileOverYMin],'FileType','text',
'Delimiter',',','OutputType','double');

    %Y Axis Orientation correction (most positive value of Y to most negative
value of Y
    if OverYMax(1,3) > OverYMax(CellNoY,3)
        OverYMax = OverYMax;
    else
        OverYMax = flip(OverYMax,1);
    end

    if OverYMin(1,3) > OverYMin(CellNoY,3)
        OverYMin = OverYMin;
    else
        OverYMin = flip(OverYMin,1);
    end

    % Collect Data in Matrix over all samples
    VxMax(:,c) = OverYMax(:,4);
    VxMin(:,c) = OverYMin(:,4);

    VyMax(:,c) = OverYMax(:,5);
    VyMin(:,c) = OverYMin(:,5);

    GAMax(:,c) = OverYMax(:,6);
    GAMin(:,c) = OverYMin(:,6);

    %Get Y-Coordinates
    YCoord(:,c) = OverYMax(:,3);


%%Sanity Check
%{
figure('name','VxMax')
plot( YCoord(:,1),VxMax(:,c))
figure('name','VxMin')
plot( YCoord(:,1),VxMin(:,c))
```

```
%}
end
%% Data Processing, Original Points Only

%X and Y Points of optimization Simulation for Plot and surface fit
PointsX=zeros(SmplNoD1*SmplNoD2,1); %X Points Of Optimization,
(X1X1X1X1X2X2X2X2 etc)
for c1x=1:SmplNoD1
    for c1y=1:SmplNoD2
PointsX((c1x-1)*SmplNoD1+c1y)=XMax-((c1x-1)*DeltaX);
    end
end

PointsY=zeros(SmplNoD1*SmplNoD2,1); %Y Points Of Optimization
(Y1Y2Y3Y4Y1Y2Y3Y etc)
for c1y=1:SmplNoD2
    for c1x=1:SmplNoD1
PointsY((c1y-1)*SmplNoD2+c1x)=DeltaYMin+((c1x-1)*DeltaDeltaY);
    end
end

PointsXMesh = zeros(1,SmplNoD1); %X Points of Optimization for mesh command
for z=1:SmplNoD1
    PointsXMesh(1,z)=XMax-((z-1)*DeltaX);
end

PointsYMesh = zeros(SmplNoD2,1); %Y Points of Optimization for mesh command
for aa=1:SmplNoD2
    PointsYMesh(aa,1)=DeltaYMin+((aa-1)*DeltaDeltaY);
end



% Generate ant print data of interest. Not Optimization related
% Average X velocity at centerline at TW hingepoint over all design points
if mod(CellNoY,2) == 0 %either one data point exactly at Y=0 or two right
next two it
    VxMaxNoZero = VxMax; %Deleting any Zero Columns for averaging in case not
all samples are used
    VxMaxNoZero( :, ~any(VxMaxNoZero,1) ) = [];
    VxMinNoZero = VxMin; %Deleting any Zero Columns for averaging in case not
all samples are used
    VxMinNoZero( :, ~any(VxMinNoZero,1) ) = [];
    VxMaxAvgTot = (mean(VxMaxNoZero,2)+mean(VxMinNoZero,2))/2;
    VxMaxAvgTot = (VxMaxAvgTot(CellNoY/2,1)+VxMaxAvgTot((CellNoY/2+1),1))/2;
else
    VxMaxNoZero = VxMax; %Deleting any Zero Columns for averaging in case not
all samples are used
    VxMaxNoZero( :, ~any(VxMaxNoZero,1) ) = [];
    VxMinNoZero = VxMin; %Deleting any Zero Columns for averaging in case not
all samples are used
    VxMinNoZero( :, ~any(VxMinNoZero,1) ) = [];
    VxMaxAvgTot = (mean(VxMaxNoZero,2)+mean(VxMinNoZero,2))/2;
    VxMaxAvgTot = (VxMaxAvgTot(ceil(CellNoY/2),1));
end
disp('Average Vx at maximum GV deflection over all design points')
disp(VxMaxAvgTot)

% Max Gust Velocity of all design points
if mod(CellNoY,2) == 0 %either one data point exactly at Y=0 or two right
next two it
```

```
    VyMaxTot = ((VyMax(CellNoY/2,:)+VyMax((CellNoY/2+1),:)-
(VyMin((CellNoY/2),:)+VyMin((CellNoY/2+1),:)))/4);
    VyMaxTot = max(VyMaxTot);
else
    VyMaxTot = ((VyMax(ceil(CellNoY/2),:)-VyMin(ceil(CellNoY/2),:))/2);
    VyMaxTot = max(VyMaxTot);
end
disp('Maximum gust velocity vy of all design points')
disp(VyMaxTot)

% Max gust velocity of each design point (pos/neg average) (C=complete)
if mod(CellNoY,2) == 0
    VyMaxC = ((VyMax(CellNoY/2,:)+VyMax((CellNoY/2+1),:)-
(VyMin((CellNoY/2),:)+VyMin((CellNoY/2+1),:)))/4);
else
    VyMaxC = ((VyMax(ceil(CellNoY/2),:)-VyMin(ceil(CellNoY/2),:))/2);
end
disp('Maximum gust velocity vy of each design point')
disp(VyMaxC)

% Max gust angle over all sample points
if mod(CellNoY,2) == 0
    GATot = ((GAMax(CellNoY/2,:)+GAMax((CellNoY/2+1),:)-
(GAMin((CellNoY/2),:)+GAMin((CellNoY/2+1),:)))/4);
    GATot = max(GATot);
else
    GATot = ((GAMax(ceil(CellNoY/2),:)+GAMin(ceil(CellNoY/2),:))/2);
    GATot = max(GATot);
end
disp('Maximum gust angle GA of all design points')
disp(GATot)

% Max gust Angle for each data Point (pos/neg average at center line)
if mod(CellNoY,2) == 0
    GAMaxC = ((GAMax(CellNoY/2,:)+GAMax((CellNoY/2+1),:)-
(GAMin((CellNoY/2),:)+GAMin((CellNoY/2+1),:)))/4);
else
    GAMaxC = ((GAMax(ceil(CellNoY/2),:)-GAMin(ceil(CellNoY/2),:))/2);
end
disp('Maximum gust angle GA of each design points')
disp(GAMaxC)


% Wake Position at each sample point (for min and Max Gust angle)
WakePointsLocMax=[];
WakePointsLocMin=[];
WakePointsLocMaxY=[];
WakePointsLocMinY=[];

WakePointsValMax=[];
WakePointsValMin=[];

for g=SampleFirst:SampleLast
    [pksWMax,locsWMax] = findpeaks(-
VxMax(:,g),'MinPeakDistance',2,'MinPeakProminence',0.5); %Find Wake for Max
Gust Angle, Tune Prominence until only wake is captured
    [pksWMin,locsWMin] = findpeaks(-
VxMin(:,g),'MinPeakDistance',2,'MinPeakProminence',0.5); %Find Wake for Min
Gust Angle, Tune Prominence until only wake is captured
    pksWMax = pksWMax*(-1);
    pksWMin = pksWMin*(-1);
```

```
    locsWMaxY(1,1) = YCoord(locsWMax(1,1),g);%write Y-coordinate of all wakes
to temporary vector (both wakes for positive and negative gv deflection)
    locsWMaxY(2,1) = YCoord(locsWMax(2,1),g);
    locsWMinY(1,1) = YCoord(locsWMin(1,1),g);
    locsWMinY(2,1) = YCoord(locsWMin(2,1),g);

    WakePointsLocMax = [WakePointsLocMax locsWMax];%add local vectors to over
all matrix (indice, y-coordinate and vx value at wake tip)
    WakePointsLocMin = [WakePointsLocMin locsWMin];
    WakePointsLocMaxY = [WakePointsLocMaxY locsWMaxY];
    WakePointsLocMinY = [WakePointsLocMinY locsWMinY];

    WakePointsValMax = [WakePointsValMax pksWMax];
    WakePointsValMin = [WakePointsValMin pksWMin];
end

%Wake Distance at each sample Point (Average of Max. negative and Max
positive)
WakeDist =  (abs(WakePointsLocMaxY(1,:) - WakePointsLocMaxY(2,:)) +
abs(WakePointsLocMinY(1,:) - WakePointsLocMinY(2,:)))/2;
disp('Distance in between wakes, average of negative and positive wake')
disp(WakeDist)

%Minimum Distance of Wake to Center of all design points
WakeDistCenter =
min([min(abs(WakePointsLocMaxY),[],1);min(abs(WakePointsLocMinY),[],1)],[],1)
-(WofWake/2); % half width of wake is substracted at this point
disp('Minimal wake distance to center, half wake width substracted')
disp(WakeDistCenter)

%Distance between two data points
DeltaY = abs(YCoord(floor((CellNoY/2+0.1)),:)-
YCoord(ceil((CellNoY/2+0.1)),:)); %Takes Y Values arround center to
callculate deltaY. Only usabel if simulation mesh is constant in area of
interes!!!
DeltaY = max(DeltaY);

%%Sanity Check
%{
figure()
plot(YCoord(:,g),VxMax(:,g))
hold on
plot(WakePointsLocMax,WakePointsValMax,'o')
hold off
figure
plot(YCoord(:,g),VxMin(:,g))
hold on
plot(WakePointsLocMin, WakePointsValMin,'o')
hold off
%}
%% Smoothing Wake for interpolation (Wake is still K.O. Criteria

%Define number of points to remove
WakeCellcount = ceil(WofWake/DeltaY);
if ~mod(WakeCellcount,2)
    WakeCellcount=WakeCellcount+1;
else
    WakeCellcount=WakeCellcount+2;
end
```

```
%remove points
VxMaxPR = VxMax;%create new matrices without the wake set to 0
VxMinPR = VxMin;

VyMaxPR = VyMax;
VyMinPR = VyMin;

GAMaxPR = GAMax;
GAMinPR = GAMin;

for h=SampleFirst:SampleLast %set wake velocity to 0
    for i=1:WakeCellcount
        VxMaxPR(WakePointsLocMax(1,h)-((WakeCellcount-1)/2)-1+i,h) = 0;
        VxMaxPR(WakePointsLocMax(2,h)-((WakeCellcount-1)/2)-1+i,h) = 0;
        VxMinPR(WakePointsLocMin(1,h)-((WakeCellcount-1)/2)-1+i,h) = 0;
        VxMinPR(WakePointsLocMin(2,h)-((WakeCellcount-1)/2)-1+i,h) = 0;

        VyMaxPR(WakePointsLocMax(1,h)-((WakeCellcount-1)/2)-1+i,h) = 0;
        VyMaxPR(WakePointsLocMax(2,h)-((WakeCellcount-1)/2)-1+i,h) = 0;
        VyMinPR(WakePointsLocMin(1,h)-((WakeCellcount-1)/2)-1+i,h) = 0;
        VyMinPR(WakePointsLocMin(2,h)-((WakeCellcount-1)/2)-1+i,h) = 0;

        GAMaxPR(WakePointsLocMax(1,h)-((WakeCellcount-1)/2)-1+i,h) = 0;
        GAMaxPR(WakePointsLocMax(2,h)-((WakeCellcount-1)/2)-1+i,h) = 0;
        GAMinPR(WakePointsLocMin(1,h)-((WakeCellcount-1)/2)-1+i,h) = 0;
        GAMinPR(WakePointsLocMin(2,h)-((WakeCellcount-1)/2)-1+i,h) = 0;
    end
end

%replace empty points with linear fit
VxMaxLF = VxMaxPR; %Create new set of matrices with linearly intermpolated
wakes
VxMinLF = VxMinPR;

VyMaxLF = VyMaxPR;
VyMinLF = VyMinPR;

GAMaxLF = GAMaxPR;
GAMinLF = GAMinPR;

for j=SampleFirst:SampleLast %set wake velocity to from zero to linearely
interpolated value
    for k=1:WakeCellcount %Generate linear delta values
        LFDeltaVxMax1 =
(VxMaxLF(WakePointsLocMax(1,j)+WakeCellcount/2+(1/2),j) -
VxMaxLF(WakePointsLocMax(1,j)-WakeCellcount/2-(1/2),j))/(WakeCellcount+1);
        LFDeltaVxMax2 =
(VxMaxLF(WakePointsLocMax(2,j)+WakeCellcount/2+(1/2),j) -
VxMaxLF(WakePointsLocMax(2,j)-WakeCellcount/2-(1/2),j))/(WakeCellcount+1);
        LFDeltaVxMin1 =
(VxMinLF(WakePointsLocMin(1,j)+WakeCellcount/2+(1/2),j) -
VxMinLF(WakePointsLocMin(1,j)-WakeCellcount/2-(1/2),j))/(WakeCellcount+1);
        LFDeltaVxMin2 =
(VxMinLF(WakePointsLocMin(2,j)+WakeCellcount/2+(1/2),j) -
VxMinLF(WakePointsLocMin(2,j)-WakeCellcount/2-(1/2),j))/(WakeCellcount+1);

        LFDeltaVyMax1 =
(VyMaxLF(WakePointsLocMax(1,j)+WakeCellcount/2+(1/2),j) -
VyMaxLF(WakePointsLocMax(1,j)-WakeCellcount/2-(1/2),j))/(WakeCellcount+1);
```

```
        LFDeltaVyMax2 =
(VyMaxLF(WakePointsLocMax(2,j)+WakeCellcount/2+(1/2),j) -
VyMaxLF(WakePointsLocMax(2,j)-WakeCellcount/2-(1/2),j))/(WakeCellcount+1);
        LFDeltaVyMin1 =
(VyMinLF(WakePointsLocMin(1,j)+WakeCellcount/2+(1/2),j) -
VyMinLF(WakePointsLocMin(1,j)-WakeCellcount/2-(1/2),j))/(WakeCellcount+1);
        LFDeltaVyMin2 =
(VyMinLF(WakePointsLocMin(2,j)+WakeCellcount/2+(1/2),j) -
VyMinLF(WakePointsLocMin(2,j)-WakeCellcount/2-(1/2),j))/(WakeCellcount+1);

        LFDeltaGAMax1 =
(GAMaxLF(WakePointsLocMax(1,j)+WakeCellcount/2+(1/2),j) -
GAMaxLF(WakePointsLocMax(1,j)-WakeCellcount/2-(1/2),j))/(WakeCellcount+1);
        LFDeltaGAMax2 =
(GAMaxLF(WakePointsLocMax(2,j)+WakeCellcount/2+(1/2),j) -
GAMaxLF(WakePointsLocMax(2,j)-WakeCellcount/2-(1/2),j))/(WakeCellcount+1);
        LFDeltaGAMin1 =
(GAMinLF(WakePointsLocMin(1,j)+WakeCellcount/2+(1/2),j) -
GAMinLF(WakePointsLocMin(1,j)-WakeCellcount/2-(1/2),j))/(WakeCellcount+1);
        LFDeltaGAMin2 =
(GAMinLF(WakePointsLocMin(2,j)+WakeCellcount/2+(1/2),j) -
GAMinLF(WakePointsLocMin(2,j)-WakeCellcount/2-(1/2),j))/(WakeCellcount+1);

        VxMaxLF(WakePointsLocMax(1,j)-((WakeCellcount-1)/2)-1+k,j) =
VxMaxLF(WakePointsLocMax(1,j)-WakeCellcount/2-(1/2),j)+k*LFDeltaVxMax1;
%Generate new data points
        VxMaxLF(WakePointsLocMax(2,j)-((WakeCellcount-1)/2)-1+k,j) =
VxMaxLF(WakePointsLocMax(2,j)-WakeCellcount/2-(1/2),j)+k*LFDeltaVxMax2;
        VxMinLF(WakePointsLocMin(1,j)-((WakeCellcount-1)/2)-1+k,j) =
VxMinLF(WakePointsLocMin(1,j)-WakeCellcount/2-(1/2),j)+k*LFDeltaVxMin1;
        VxMinLF(WakePointsLocMin(2,j)-((WakeCellcount-1)/2)-1+k,j) =
VxMinLF(WakePointsLocMin(2,j)-WakeCellcount/2-(1/2),j)+k*LFDeltaVxMin2;

        VyMaxLF(WakePointsLocMax(1,j)-((WakeCellcount-1)/2)-1+k,j) =
VyMaxLF(WakePointsLocMax(1,j)-WakeCellcount/2-(1/2),j)+k*LFDeltaVyMax1;
        VyMaxLF(WakePointsLocMax(2,j)-((WakeCellcount-1)/2)-1+k,j) =
VyMaxLF(WakePointsLocMax(2,j)-WakeCellcount/2-(1/2),j)+k*LFDeltaVyMax2;
        VyMinLF(WakePointsLocMin(1,j)-((WakeCellcount-1)/2)-1+k,j) =
VyMinLF(WakePointsLocMin(1,j)-WakeCellcount/2-(1/2),j)+k*LFDeltaVyMin1;
        VyMinLF(WakePointsLocMin(2,j)-((WakeCellcount-1)/2)-1+k,j) =
VyMinLF(WakePointsLocMin(2,j)-WakeCellcount/2-(1/2),j)+k*LFDeltaVyMin2;

        GAMaxLF(WakePointsLocMax(1,j)-((WakeCellcount-1)/2)-1+k,j) =
GAMaxLF(WakePointsLocMax(1,j)-WakeCellcount/2-(1/2),j)+k*LFDeltaGAMax1;
        GAMaxLF(WakePointsLocMax(2,j)-((WakeCellcount-1)/2)-1+k,j) =
GAMaxLF(WakePointsLocMax(2,j)-WakeCellcount/2-(1/2),j)+k*LFDeltaGAMax2;
        GAMinLF(WakePointsLocMin(1,j)-((WakeCellcount-1)/2)-1+k,j) =
GAMinLF(WakePointsLocMin(1,j)-WakeCellcount/2-(1/2),j)+k*LFDeltaGAMin1;
        GAMinLF(WakePointsLocMin(2,j)-((WakeCellcount-1)/2)-1+k,j) =
GAMinLF(WakePointsLocMin(2,j)-WakeCellcount/2-(1/2),j)+k*LFDeltaGAMin2;
    end
end

%% Data Interpolation

if (SampleFirst ~= 1) || (SampleLast ~= SmplNoD1*SmplNoD2)
    disp('Data Set incomplete, No Interpolation done')
else

    %%Interpolation 1 (Y-Direction), Using Smoothed Data (without wake)
```

```
    %Position Points (Usable for whole Interpolation 1)
    DeltaGV =[]; %Generation of Design Point Vectors, distance between gust
vanes
    for a=1:SmplNoD2
        DeltaGVAdd  = zeros(CellNoY,1);
        DeltaGVAdd(:,1)=DeltaYMin+DeltaDeltaY*(a-1);
        DeltaGV = [DeltaGV;DeltaGVAdd]; %collection Matrix
    end

    YPosComp=[]; %Collecting of y coord vectros assosciated with design
points
    for b=1:SmplNoD1
        YPos =[];
        for d=1:SmplNoD2
            YPosAdd  = zeros(CellNoY,1);
            YPosAdd(:,1) = YCoord(:,((b-1)*SmplNoD1)+d);
            YPos = [YPos;YPosAdd];
        end
        YPosComp = [YPosComp YPos];
    end

    %Data Values
    VxPComp=[];
    VxNComp=[];

    VyPComp=[];
    VyNComp=[];

    GAPComp=[];
    GANComp=[];

    for e=1:SmplNoD1 %Generation of Vectors (P=Positive, N=Negative) needed
for interpolation
        VxP =[]; %Generating of Vx
        VxN =[];

        VyP =[]; %Generation of Vy
        VyN =[];

        GAP =[]; %Generation of GA
        GAN =[];

        for f=1:SmplNoD2 %rearanging smoothed data, result: 4 columns, each
column contains the data of one x position, needed in this form for
interpolation
            VxPAdd  = zeros(CellNoY,1);
            VxPAdd(:,1) = VxMaxLF(:,((e-1)*SmplNoD1)+f);
            VxP = [VxP;VxPAdd];
            VxNAdd  = zeros(CellNoY,1);
            VxNAdd(:,1) = VxMinLF(:,((e-1)*SmplNoD1)+f);
            VxN = [VxN;VxNAdd];

            VyPAdd  = zeros(CellNoY,1);
            VyPAdd(:,1) = VyMaxLF(:,((e-1)*SmplNoD1)+f);
            VyP = [VyP;VyPAdd];
            VyNAdd  = zeros(CellNoY,1);
            VyNAdd(:,1) = VyMinLF(:,((e-1)*SmplNoD1)+f);
            VyN = [VyN;VyNAdd];

            GAPAdd  = zeros(CellNoY,1);
            GAPAdd(:,1) = GAMaxLF(:,((e-1)*SmplNoD1)+f);
```

```
            GAP = [GAP;GAPAdd];
            GANAdd  = zeros(CellNoY,1);
            GANAdd(:,1) = GAMinLF(:,((e-1)*SmplNoD1)+f);
            GAN = [GAN;GANAdd];
        end
        VxPComp = [VxPComp VxP];
        VxNComp = [VxNComp VxN];

        VyPComp = [VyPComp VyP];
        VyNComp = [VyNComp VyN];

        GAPComp = [GAPComp GAP];
        GANComp = [GANComp GAN];
    end

    VxMaxI1 = [];
    VxMinI1 = [];

    VyMaxI1 = [];
    VyMinI1 = [];

    GAMaxI1 = [];
    GAMinI1 = [];

    YCoordI1 = [];

    %Interpolation 1
    for  l=1:SmplNoD1
        [DeltaGVI,YPosI] =
meshgrid((DeltaYMin:(DeltaDeltaY/(IPNo+1)):DeltaYMin+(SmplNoD2-
1)*DeltaDeltaY),YCoord(:,(l*SmplNoD1)-SmplNoD1+1));% Generation of desired
data points, !!!YCoord assumed to be constant!!!

        VxIp =
griddata(DeltaGV,YPosComp(:,l),VxPComp(:,l),DeltaGVI,YPosI,'cubic'); %
Generation of  interpolated values for desired design points
        VxIn =
griddata(DeltaGV,YPosComp(:,l),VxNComp(:,l),DeltaGVI,YPosI,'cubic');

        VyIp =
griddata(DeltaGV,YPosComp(:,l),VyPComp(:,l),DeltaGVI,YPosI,'cubic');
        VyIn =
griddata(DeltaGV,YPosComp(:,l),VyNComp(:,l),DeltaGVI,YPosI,'cubic');

        GAIp =
griddata(DeltaGV,YPosComp(:,l),GAPComp(:,l),DeltaGVI,YPosI,'cubic');
        GAIn =
griddata(DeltaGV,YPosComp(:,l),GANComp(:,l),DeltaGVI,YPosI,'cubic');

        VxMaxI1 = [VxMaxI1 VxIp];
        VxMinI1 = [VxMinI1 VxIn];

        VyMaxI1 = [VyMaxI1 VyIp];
        VyMinI1 = [VyMinI1 VyIn];

        GAMaxI1 = [GAMaxI1 GAIp];
        GAMinI1 = [GAMinI1 GAIn];
```

```
%%Graphical SanityCheck
        %{
        figure()
        plot3(DeltaGVI,YPosI,VxMaxI1(:,(((l-
1)*SmplNoD2New)+1):(l*SmplNoD2New)), 'Marker','o' )
        hold on
        mesh(DeltaGVI,YPosI,VxMaxI1(:,(((l-
1)*SmplNoD2New)+1):(l*SmplNoD2New)))
        hold off

        figure()
        plot3(DeltaGVI,YPosI,VxMinI1(:,(((l-
1)*SmplNoD2New)+1):(l*SmplNoD2New)), 'Marker','o' )
        hold on
        mesh(DeltaGVI,YPosI,VxMinI1(:,(((l-
1)*SmplNoD2New)+1):(l*SmplNoD2New)))
        hold off

        figure()
        plot3(DeltaGVI,YPosI,VyMaxI1(:,(((l-
1)*SmplNoD2New)+1):(l*SmplNoD2New)), 'Marker','o' )
        hold on
        mesh(DeltaGVI,YPosI,VyMaxI1(:,(((l-
1)*SmplNoD2New)+1):(l*SmplNoD2New)))
        hold off

        figure()
        plot3(DeltaGVI,YPosI,VyMinI1(:,(((l-
1)*SmplNoD2New)+1):(l*SmplNoD2New)), 'Marker','o' )
        hold on
        mesh(DeltaGVI,YPosI,VyMinI1(:,(((l-
1)*SmplNoD2New)+1):(l*SmplNoD2New)))
        hold off

        figure()
        plot3(DeltaGVI,YPosI,GAMaxI1(:,(((l-
1)*SmplNoD2New)+1):(l*SmplNoD2New)), 'Marker','o' )
        hold on
        mesh(DeltaGVI,YPosI,GAMaxI1(:,(((l-
1)*SmplNoD2New)+1):(l*SmplNoD2New)))
        hold off

        figure()
        plot3(DeltaGVI,YPosI,GAMinI1(:,(((l-
1)*SmplNoD2New)+1):(l*SmplNoD2New)), 'Marker','o' )
        hold on
        mesh(DeltaGVI,YPosI,GAMinI1(:,(((l-
1)*SmplNoD2New)+1):(l*SmplNoD2New)))
        hold off
        %}
    end

    for m=1:SmplNoD1 %Generaing matching YCoord for interpolated values,
arranged in original form X1Y1X1Y2X1Y3....XnY1XnY2 etc
        for n=1:SmplNoD2New
            YCoordI1(:,(m-1)*SmplNoD2New+n)=YCoord(:,(m-
1)*SmplNoD2+ceil(n/(IPNo+1)));
        end
    end
```

```
    %%Interpolation 2 (X-Direction), Using Smoothed Data (without wake)

    %Position Points (Usable for whole Interpolation 2)
    XGV =[]; %Generation of Design Point Vector, stream wise position
    for n=1:SmplNoD1
        XGVAdd  = zeros(CellNoY,1);
        XGVAdd(:,1)=XMax-DeltaX*(n-1);
        XGV = [XGV;XGVAdd];
    end

    YPosComp2=[]; %Collecting of y coord vectros assosciated with design
points
    for o=1:SmplNoD2New
        YPos2 =[];
        for p=1:SmplNoD1
            YPosAdd2  = zeros(CellNoY,1);
            YPosAdd2(:,1) = YCoordI1(:,o+(p-1)*SmplNoD2New); %% Generate new
YCoord with now 4*(4+3*7) = 100 Columns
            YPos2 = [YPos2;YPosAdd2];
        end
        YPosComp2 = [YPosComp2 YPos2];
    end

    %Data Values
    VxPComp2=[];
    VxNComp2=[];

    VyPComp2=[];
    VyNComp2=[];

    GAPComp2=[];
    GANComp2=[];

    for q=1:SmplNoD2New %Generation of Vectors (P=Positive, N=Negative)
needed for interpolation
        VxP2 =[]; %Generation of Vx
        VxN2 =[];

        VyP2 =[]; %Generation of Vy
        VyN2 =[];

        GAP2 =[]; %Generation of GA
        GAN2 =[];

        for r=1:SmplNoD1%rearanging smoothed data, result: SmplNoD2New
columns, each column contains the data of one (new) y position, needed in
this form for interpolation
            VxPAdd2  = zeros(CellNoY,1);
            VxPAdd2(:,1) = VxMaxI1(:,q+SmplNoD2New*(r-1));
            VxP2 = [VxP2;VxPAdd2];
            VxNAdd2  = zeros(CellNoY,1);
            VxNAdd2(:,1) = VxMinI1(:,q+SmplNoD2New*(r-1));
            VxN2 = [VxN2;VxNAdd2];

            VyPAdd2  = zeros(CellNoY,1);
            VyPAdd2(:,1) = VyMaxI1(:,q+SmplNoD2New*(r-1));
            VyP2 = [VyP2;VyPAdd2];
            VyNAdd2  = zeros(CellNoY,1);
            VyNAdd2(:,1) = VyMinI1(:,q+SmplNoD2New*(r-1));
```

```
            VyN2 = [VyN2;VyNAdd2];

            GAPAdd2  = zeros(CellNoY,1);
            GAPAdd2(:,1) = GAMaxI1(:,q+SmplNoD2New*(r-1));
            GAP2 = [GAP2;GAPAdd2];
            GANAdd2  = zeros(CellNoY,1);
            GANAdd2(:,1) = GAMinI1(:,q+SmplNoD2New*(r-1));
            GAN2 = [GAN2;GANAdd2];
        end
        VxPComp2 = [VxPComp2 VxP2];
        VxNComp2 = [VxNComp2 VxN2];

        VyPComp2 = [VyPComp2 VyP2];
        VyNComp2 = [VyNComp2 VyN2];

        GAPComp2 = [GAPComp2 GAP2];
        GANComp2 = [GANComp2 GAN2];
    end

    VxMaxI2 = [];
    VxMinI2 = [];

    VyMaxI2 = [];
    VyMinI2 = [];

    GAMaxI2 = [];
    GAMinI2 = [];

    YCoordI2 = [];

    %Interpolation 2
    for  s=1:SmplNoD2New
        [XGVI,YPosI2] = meshgrid((XMax:(-(DeltaX/(IPNo+1))):XMax-(SmplNoD1-
1)*DeltaX),YCoordI1(:,s));% Generation of desired data points, !!!YCoord
assumed to be constant!!!

        VxIp2 =
griddata(XGV,YPosComp2(:,s),VxPComp2(:,s),XGVI,YPosI2,'cubic'); % Generation
of values for desired data points
        VxIn2 =
griddata(XGV,YPosComp2(:,s),VxNComp2(:,s),XGVI,YPosI2,'cubic');

        VyIp2 =
griddata(XGV,YPosComp2(:,s),VyPComp2(:,s),XGVI,YPosI2,'cubic');
        VyIn2 =
griddata(XGV,YPosComp2(:,s),VyNComp2(:,s),XGVI,YPosI2,'cubic');

        GAIp2 =
griddata(XGV,YPosComp2(:,s),GAPComp2(:,s),XGVI,YPosI2,'cubic');
        GAIn2 =
griddata(XGV,YPosComp2(:,s),GANComp2(:,s),XGVI,YPosI2,'cubic');

        VxMaxI2 = [VxMaxI2 VxIp2];
        VxMinI2 = [VxMinI2 VxIn2];

        VyMaxI2 = [VyMaxI2 VyIp2];
        VyMinI2 = [VyMinI2 VyIn2];

        GAMaxI2 = [GAMaxI2 GAIp2];
        GAMinI2 = [GAMinI2 GAIn2];
```

```
        %Graphical SanityCheck
        %{
        figure()
        plot3(XGVI,YPosI2,VxMaxI2(:,(((s-1)*SmplNoD1New)+1):(s*SmplNoD1New)),
'Marker','o' )
        hold on
        mesh(XGVI,YPosI2,VxMaxI2(:,(((s-1)*SmplNoD1New)+1):(s*SmplNoD1New)))
        hold off

        figure()
        plot3(XGVI,YPosI2,VxMinI2(:,(((s-1)*SmplNoD1New)+1):(s*SmplNoD1New)),
'Marker','o' )
        hold on
        mesh(XGVI,YPosI2,VxMinI2(:,(((s-1)*SmplNoD1New)+1):(s*SmplNoD1New)))
        hold off

        figure()
        plot3(XGVI,YPosI2,VyMaxI2(:,(((s-1)*SmplNoD1New)+1):(s*SmplNoD1New)),
'Marker','o' )
        hold on
        mesh(XGVI,YPosI2,VyMaxI2(:,(((s-1)*SmplNoD1New)+1):(s*SmplNoD1New)))
        hold off

        figure()
        plot3(XGVI,YPosI2,VyMinI2(:,(((s-1)*SmplNoD1New)+1):(s*SmplNoD1New)),
'Marker','o' )
        hold on
        mesh(XGVI,YPosI2,VyMinI2(:,(((s-1)*SmplNoD1New)+1):(s*SmplNoD1New)))
        hold off

        figure()
        plot3(XGVI,YPosI2,GAMaxI2(:,(((s-1)*SmplNoD1New)+1):(s*SmplNoD1New)),
'Marker','o' )
        hold on
        mesh(XGVI,YPosI2,GAMaxI2(:,(((s-1)*SmplNoD1New)+1):(s*SmplNoD1New)))
        hold off

        figure()
        plot3(XGVI,YPosI2,GAMinI2(:,(((s-1)*SmplNoD1New)+1):(s*SmplNoD1New)),
'Marker','o' )
        hold on
        mesh(XGVI,YPosI2,GAMinI2(:,(((s-1)*SmplNoD1New)+1):(s*SmplNoD1New)))
        hold off
        %}
    end

    for t=1:SmplNoD2New %Generaing matching YCoord for interpolated values,
arranged in original form X1Y1X1Y2X1Y3....XnY1XnY2 etc
            for u=1:SmplNoD1New
            YCoordI2(:,(t-1)*SmplNoD2New+u)=YCoordI1(:,t+(ceil(u/(IPNo+1))-
1)*SmplNoD1New);
            end
        end

    %Rearranging Matrixes to Y1X1Y2X1Y3X2....Y1X3Y2X3Y3X3..., Final Matrix
    for v=1:SmplNoD1New
        for w=1:SmplNoD2New
        VxMaxIF(:,(v-1)*SmplNoD1New+w) = VxMaxI2(:,(w-1)*SmplNoD2New+v);
        VxMinIF(:,(v-1)*SmplNoD1New+w) = VxMinI2(:,(w-1)*SmplNoD2New+v);
```

```
        VyMaxIF(:,(v-1)*SmplNoD1New+w) = VyMaxI2(:,(w-1)*SmplNoD2New+v);
        VyMinIF(:,(v-1)*SmplNoD1New+w) = VyMinI2(:,(w-1)*SmplNoD2New+v);

        GAMaxIF(:,(v-1)*SmplNoD1New+w) = GAMaxI2(:,(w-1)*SmplNoD2New+v);
        GAMinIF(:,(v-1)*SmplNoD1New+w) = GAMinI2(:,(w-1)*SmplNoD2New+v);

        YCoordIF(:,(v-1)*SmplNoD1New+w) = YCoordI2(:,(w-1)*SmplNoD2New+v);
        end
    end


    %%Sanity Check
    %{
    for x=1:SmplNoD2New
        figure()
        plot3(DeltaGVI,YPosI,VxMaxIF(:,(((x-
1)*SmplNoD2New)+1):(x*SmplNoD2New)), 'Marker','o' )
        hold on
        mesh(DeltaGVI,YPosI,VxMaxIF(:,(((x-
1)*SmplNoD2New)+1):(x*SmplNoD2New)))
        hold off

        figure()
        plot3(DeltaGVI,YPosI,VxMinIF(:,(((x-
1)*SmplNoD2New)+1):(x*SmplNoD2New)), 'Marker','o' )
        hold on
        mesh(DeltaGVI,YPosI,VxMinIF(:,(((x-
1)*SmplNoD2New)+1):(x*SmplNoD2New)))
        hold off
        figure()
        plot3(DeltaGVI,YPosI,VyMaxIF(:,(((x-
1)*SmplNoD2New)+1):(x*SmplNoD2New)), 'Marker','o' )
        hold on
        mesh(DeltaGVI,YPosI,VyMaxIF(:,(((x-
1)*SmplNoD2New)+1):(x*SmplNoD2New)))
        hold off
        figure()
        plot3(DeltaGVI,YPosI,VyMinIF(:,(((x-
1)*SmplNoD2New)+1):(x*SmplNoD2New)), 'Marker','o' )
        hold on
        mesh(DeltaGVI,YPosI,VyMinIF(:,(((x-
1)*SmplNoD2New)+1):(x*SmplNoD2New)))
        hold off

        figure()
        plot3(DeltaGVI,YPosI,GAMaxIF(:,(((x-
1)*SmplNoD2New)+1):(x*SmplNoD2New)), 'Marker','o' )
        hold on
        mesh(DeltaGVI,YPosI,GAMaxIF(:,(((x-
1)*SmplNoD2New)+1):(x*SmplNoD2New)))
        hold off

        figure()
        plot3(DeltaGVI,YPosI,GAMinIF(:,(((x-
1)*SmplNoD2New)+1):(x*SmplNoD2New)), 'Marker','o' )
        hold on
        mesh(DeltaGVI,YPosI,GAMinIF(:,(((x-
1)*SmplNoD2New)+1):(x*SmplNoD2New)))
        hold off
    end
    %}
```

```
end
%% Data Aquisition form Interpolatet Data (Gust Deviation & Vy Deviation)

%Cellcount for minimal necesary Wake dist to Y=0, as given by test section
WtoCMinCC=ceil(WToCMin/DeltaY);

%Shortend Vectors (To Area of interes arround Y=0)
if mod(CellNoY,2) == 0
CellCToremove = CellNoY/2-WtoCMinCC;
else
CellCToremove = (CellNoY-1)/2-WtoCMinCC;
end

VyMaxIFS = VyMaxIF; %Renaming Matrices
VyMinIFS = VyMinIF;

GAMaxIFS = GAMaxIF;
GAMinIFS = GAMinIF;

YCoordIFS = YCoordIF;

VyMaxIFS(1:CellCToremove,:) = [];
VyMaxIFS(CellNoY-2*CellCToremove+1:CellNoY-CellCToremove,:) = [];
VyMinIFS(1:CellCToremove,:) = [];
VyMinIFS(CellNoY-2*CellCToremove+1:CellNoY-CellCToremove,:) = [];

GAMaxIFS(1:CellCToremove,:) = [];
GAMaxIFS(CellNoY-2*CellCToremove+1:CellNoY-CellCToremove,:) = [];
GAMinIFS(1:CellCToremove,:) = [];
GAMinIFS(CellNoY-2*CellCToremove+1:CellNoY-CellCToremove,:) = [];

YCoordIFS(1:CellCToremove,:) = [];
YCoordIFS(CellNoY-2*CellCToremove+1:CellNoY-CellCToremove,:) = [];

[DeltaGVIFS,YPosIFS] =
meshgrid((DeltaYMin:(DeltaDeltaY/(IPNo+1)):DeltaYMin+(SmplNoD2-
1)*DeltaDeltaY),YCoordIFS(:,(l*SmplNoD1)-SmplNoD1+1));

%%Sanity Check
    %{
    for y=1:SmplNoD2New

        figure()
        plot3(DeltaGVIFS,YPosIFS,VyMaxIFS(:,(((y-
1)*SmplNoD2New)+1):(y*SmplNoD2New)), 'Marker','o' )
        hold on
        mesh(DeltaGVIFS,YPosIFS,VyMaxIFS(:,(((y-
1)*SmplNoD2New)+1):(y*SmplNoD2New)))
        hold off

        figure()
        plot3(DeltaGVIFS,YPosIFS,VyMinIFS(:,(((y-
1)*SmplNoD2New)+1):(y*SmplNoD2New)), 'Marker','o' )
        hold on
        mesh(DeltaGVIFS,YPosIFS,VyMinIFS(:,(((y-
1)*SmplNoD2New)+1):(y*SmplNoD2New)))
        hold off

        figure()
```

```
        plot3(DeltaGVIFS,YPosIFS,GAMaxIFS(:,(((y-
1)*SmplNoD2New)+1):(y*SmplNoD2New)), 'Marker','o' )
        hold on
        mesh(DeltaGVIFS,YPosIFS,GAMaxIFS(:,(((y-
1)*SmplNoD2New)+1):(y*SmplNoD2New)))
        hold off

        figure()
        plot3(DeltaGVIFS,YPosIFS,GAMinIFS(:,(((y-
1)*SmplNoD2New)+1):(y*SmplNoD2New)), 'Marker','o' )
        hold on
        mesh(DeltaGVIFS,YPosIFS,GAMinIFS(:,(((y-
1)*SmplNoD2New)+1):(y*SmplNoD2New)))
        hold off
    end
    %}

%Average Values
VyMinMaxAvg = (mean(abs(VyMaxIFS),1)+mean(abs(VyMinIFS),1))/2;

%VyMinAvg = mean(abs(VyMinIFS),1); %for deviation only in relation to neg or
positive gust
%VyMaxAvg = mean(abs(VyMaxIFS),1);

GAMinMaxAvg = (mean(abs(GAMaxIFS),1)+mean(abs(GAMinIFS),1))/2;

%GAMinAvg = mean(abs(GAMinIFS),1); %for deviation only in relation to neg or
positive gust
%GAMaxAvg = mean(abs(GAMaxIFS),1);

%%Initial wrong average calculation
%{
%%Wrong Averaging: Averaging before deviation instead of after
%Mean Absolute deviation normalized with avg max Gust angle/ avg max Vy
VyMinMaxIFS = (abs(VyMaxIFS)+abs(VyMinIFS))/2; %Averaging over Pos and Neg
Gust Values
GAMinMaxIFS = (abs(GAMaxIFS)+abs(GAMinIFS))/2;

VyDev = mean(abs(VyMinMaxIFS - VyMinMaxAvg),1); %Mean Absolut Deviation
GADev = mean(abs(GAMinMaxIFS - GAMinMaxAvg),1);
%}

%Vy and GA deviation calculation, deviation calculated with average pos and
neg gust
VyDev = mean((abs(abs(VyMaxIFS)-VyMinMaxAvg)+abs(abs(VyMinIFS)-
VyMinMaxAvg)/2),1);
GADev = mean((abs(abs(GAMaxIFS)-GAMinMaxAvg)+abs(abs(GAMinIFS)-
GAMinMaxAvg)/2),1);

%%for deviation only in relation to neg or positive gust
%{
VyDev = mean((abs(abs(VyMaxIFS)-VyMaxAvg)+abs(abs(VyMinIFS)-VyMinAvg)/2),1);
GADev = mean((abs(abs(GAMaxIFS)-GAMaxAvg)+abs(abs(GAMinIFS)-GAMinAvg)/2),1);
%}
VyDevNorm = VyDev./VyMinMaxAvg; %Norming the deviation with the average max
velocity at design point
GADevNorm = GADev./GAMinMaxAvg;

%% Plotting Data
```

```
%Max Gust Velocity, original data
MaxGVM = zeros(SmplNoD2,SmplNoD1);
for ab=1:SmplNoD1
    for ac=1:SmplNoD2
       MaxGVM(ac,ab)= VyMaxC(1,(ab-1)*SmplNoD1+ac);
    end
end

figure()
mesh(PointsXMesh,PointsYMesh,MaxGVM)
hold on
MaxGVFit = fit([PointsX,PointsY],transpose(VyMaxC),'poly33');
plot(MaxGVFit)
plot3(PointsX,PointsY,transpose(VyMaxC),'o')
hold off

%Max Gust Angle, original data
MaxGAM = zeros(SmplNoD2,SmplNoD1);
for ad=1:SmplNoD1
    for ae=1:SmplNoD2
       MaxGAM(ae,ad)= GAMaxC(1,(ad-1)*SmplNoD1+ae);
    end
end

figure()
mesh(PointsXMesh,PointsYMesh,MaxGAM)
hold on
MaxGAFit = fit([PointsX,PointsY],transpose(GAMaxC),'poly33');
plot(MaxGAFit)
plot3(PointsX,PointsY,transpose(GAMaxC),'o')
hold off

%Wake Dist, original data
WakeDistM = zeros(SmplNoD2,SmplNoD1);
for af=1:SmplNoD1
    for ag=1:SmplNoD2
       WakeDistM(ag,af)= WakeDist(1,(af-1)*SmplNoD1+ag);
    end
end

figure()
mesh(PointsXMesh,PointsYMesh,WakeDistM)
hold on
plot(fit([PointsX,PointsY],transpose(WakeDist),'poly33'))
plot3(PointsX,PointsY,transpose(WakeDist),'o')
hold off

%Min Wake Dist to Y=0, original data
WakeDistCenterM = zeros(SmplNoD2,SmplNoD1);
for ah=1:SmplNoD1
    for ai=1:SmplNoD2
       WakeDistCenterM(ai,ah)= WakeDistCenter(1,(ah-1)*SmplNoD1+ai);
    end
end

figure()
mesh(PointsXMesh,PointsYMesh,WakeDistCenterM)
hold on
WakeDistCenterFit =
fit([PointsX,PointsY],transpose(WakeDistCenter),'poly33');
plot(WakeDistCenterFit)
```

```
plot3(PointsX,PointsY,transpose(WakeDistCenter),'o')
hold off

%Mean Absolute Deviation (Nomralized with averaged Max Gust Angle/Velocity)
[XGVIMgrid,DeltaGVIMGrid]=meshgrid(XGVI(1,:),DeltaGVI(1,:)); %Generating
Points for Mesh
XGVIMesh=XGVIMgrid(1,:);
DeltaGVIMesh=DeltaGVIMGrid(:,1);

DeltaGVIP = []; %Generating position points for plot, gust vane Spacing
for aj=1:SmplNoD2New
DeltaGVIP = [DeltaGVIP;DeltaGVIMesh];
end

XGVIMeshTP = transpose(XGVIMesh); %Generating points for plot, gust vane
stream wise position
XGVIP = zeros(SmplNoD2*SmplNoD2,1);
for ak=1:SmplNoD1New
    for al=1:SmplNoD2New
    XGVIP((ak-1)*SmplNoD1New+al,1)= XGVIMeshTP(ak,1);
    end
end

VyDevNormM = zeros(SmplNoD2New,SmplNoD1New); %Generating points for plot,
Mean Absolute Deviation, Velocity
for am=1:SmplNoD1New
    for an=1:SmplNoD2New
       VyDevNormM(an,am)= VyDevNorm(1,(am-1)*SmplNoD1New+an);
    end
end

figure()
mesh(XGVIMesh,DeltaGVIMesh,VyDevNormM)
hold on
%plot(fit([XGVIP,DeltaGVIP],transpose(VyDevNorm),'cubicinterp'))
plot3(XGVIP,DeltaGVIP,transpose(VyDevNorm),'o')
hold off

GADevNormM = zeros(SmplNoD2New,SmplNoD1New); %Generating points for plot,
Mean Absolute Deviation, gust angle
for ao=1:SmplNoD1New
    for ap=1:SmplNoD2New
       GADevNormM(ap,ao)= GADevNorm(1,(ao-1)*SmplNoD1New+ap);
    end
end

figure()
mesh(XGVIMesh,DeltaGVIMesh,GADevNormM)
hold on
%plot(fit([XGVIP,DeltaGVIP],transpose(GADevNorm),'cubicinterp'))
plot3(XGVIP,DeltaGVIP,transpose(GADevNorm),'o')
hold off

%% Interpolate VxMax, GAMax and Wake dist to Center Data

MaxGVIM=zeros(SmplNoD2New,SmplNoD1New);%fitted directly to original data, not
interpolated data, for increased accuracy
MaxGAIM=zeros(SmplNoD2New,SmplNoD1New);
WakeDistCenterIM=zeros(SmplNoD2New,SmplNoD1New);
for aq=1:SmplNoD1New
    for ar=1:SmplNoD2New
```

```
        MaxGVIM(ar,aq)= MaxGVFit(XGVIMesh(1,aq),DeltaGVIMesh(ar,1));%MaxGV
        MaxGAIM(ar,aq)= MaxGAFit(XGVIMesh(1,aq),DeltaGVIMesh(ar,1));%MaxGA
        WakeDistCenterIM(ar,aq)=
WakeDistCenterFit(XGVIMesh(1,aq),DeltaGVIMesh(ar,1));
    end
end

%%Sanity Check
%{
figure()
mesh(XGVIMesh,DeltaGVIMesh, MaxGVIM)
figure()
mesh(XGVIMesh,DeltaGVIMesh, MaxGAIM)
figure()
mesh(XGVIMesh,DeltaGVIMesh, WakeDistCenterIM)
%}
%% Cropp Data due to Boundary Conditions

WakeDistCenterCrM=WakeDistCenterIM; %set all wake dist values to zero if to
small
for as=1:SmplNoD1New
    WToCDel=1;
    while WakeDistCenterCrM(WToCDel,as) < WToCMin
          WakeDistCenterCrM(WToCDel,as) = 0;
          WToCDel= WToCDel+1;
    end
end

%Cropping due to TToEndMin
TToEDel=0; %set all design point coordinates to zero if they are to close to
the test section
while XGVIMesh(1,SmplNoD1New-TToEDel) < ((TToEndMin+GVc)*1000) %(given in m
but formula works in mm)
    TToEDel=TToEDel+1;
end

%Cropping of VxMax, GAMax, GADevNormM and VyDevNormM

VyMaxCrM = MaxGVIM;
GAMaxCrM = MaxGAIM;
VyDevNormCrM = VyDevNormM;
GADevNormCrM = GADevNormM;
XGVICrMesh = XGVIMesh;

for at=1:SmplNoD1New
    WToCDel=1;
    while WakeDistCenterCrM(WToCDel,at) == 0
          VyMaxCrM(WToCDel,at) = 0;
          GAMaxCrM(WToCDel,at) = 0;
          VyDevNormCrM(WToCDel,at) = 0;
          GADevNormCrM(WToCDel,at) = 0;

          WToCDel= WToCDel+1;
    end
end

VyMaxCrM(:,(SmplNoD1New-TToEDel+1):SmplNoD1New)=[];
% VyMaxCrM(1:WToCDel,:)=[];

GAMaxCrM(:,(SmplNoD1New-TToEDel+1):SmplNoD1New)=[];
% GAMaxCrM(1:WToCDel,:)=[];
```

```
VyDevNormCrM(:,(SmplNoD1New-TToEDel+1):SmplNoD1New)=[];
% VyDevNormCrM(1:WToCDel,:)=[];

GADevNormCrM(:,(SmplNoD1New-TToEDel+1):SmplNoD1New)=[];
% GADevNormCrM(1:WToCDel,:)=[];

XGVICrMesh(:,(SmplNoD1New-TToEDel+1):SmplNoD1New)=[];

figure('Name','Cropped VyMax')
mesh(XGVICrMesh,DeltaGVIMesh,VyMaxCrM)

figure('Name','Cropped GAMax')
mesh(XGVICrMesh,DeltaGVIMesh,GAMaxCrM)

figure('Name','Cropped VyMax Deviation')
mesh(XGVICrMesh,DeltaGVIMesh,VyDevNormCrM)

figure('Name','Cropped GAMax Deviation')
mesh(XGVICrMesh,DeltaGVIMesh,GADevNormCrM)

%% Find Optimum With Weights

% Normalize Data (Spread between 0 and 1, = beeing worst, 1 beeing best)
VyMaxCrMNorm = VyMaxCrM-min(VyMaxCrM(VyMaxCrM>0)); %Shift Values so lowest is
at 0
VyMaxCrMNorm = VyMaxCrMNorm/max(max(VyMaxCrMNorm)); %Normalize with biggest
value

GAMaxCrMNorm = GAMaxCrM-min(GAMaxCrM(GAMaxCrM>0)); %Shift values so lowest is
at 0
GAMaxCrMNorm = GAMaxCrMNorm/max(max(GAMaxCrMNorm)); %Normalize with biggest
value

VyDev2NormCrM = max(max(VyDevNormCrM))-VyDevNormCrM; %Substract values from
biggest deviation --> biggest deviation is zero, smallest deviation is equal
to biggest value
GADev2NormCrM = max(max(GADevNormCrM))-GADevNormCrM;

for au=1:SmplNoD1New-TToEDel % values set to zero in pervious step are now 1
and need to be set to 0 agian
    InfTo0=1;
    while WakeDistCenterCrM(InfTo0,au) == 0
          VyDev2NormCrM(InfTo0,au) = 0;
          GADev2NormCrM(InfTo0,au) = 0;
          VyMaxCrMNorm(InfTo0,au) = 0;
          GAMaxCrMNorm(InfTo0,au) = 0;

          InfTo0= InfTo0+1;
    end
end

VyDev2NormCrM = VyDev2NormCrM/max(max(VyDev2NormCrM)); %spread values between
1 and 0
GADev2NormCrM = GADev2NormCrM/max(max(GADev2NormCrM));

figure('Name','Cropped, normed VyMax')
mesh(XGVICrMesh,DeltaGVIMesh,VyMaxCrMNorm)
```

```
figure('Name','Cropped, normed GAMax')
mesh(XGVICrMesh,DeltaGVIMesh,GAMaxCrMNorm)

figure('Name','Cropped, normed VyMax Deviation')
mesh(XGVICrMesh,DeltaGVIMesh,VyDev2NormCrM)

figure('Name','Cropped, normed GAMax Deviation')
mesh(XGVICrMesh,DeltaGVIMesh,GADev2NormCrM)


% Add Up Data With Weights
GAWeight = GAMaxCrMNorm*WVGAmax/(WVGAmax+WVGADev) +
GADev2NormCrM*WVGADev/(WVGAmax+WVGADev);
VyWeight = VyMaxCrMNorm*WVGAmax/(WVGAmax+WVGADev) +
VyDev2NormCrM*WVGADev/(WVGAmax+WVGADev);

figure('Name',['Weighted Vy Fuction, Weights (VGAMax and VGADev): '
num2str(WVGAmax/(WVGAmax+WVGADev)) ' and '
num2str(WVGADev/(WVGAmax+WVGADev))])
mesh(XGVICrMesh,DeltaGVIMesh,VyWeight)

figure('Name',['Weighted GA Fuction, Weights (VGAMax and VGADev): '
num2str(WVGAmax/(WVGAmax+WVGADev)) ' and '
num2str(WVGADev/(WVGAmax+WVGADev))])
mesh(XGVICrMesh,DeltaGVIMesh,GAWeight)

% Find Max Value
[GAWeightMax,GAWeightCol] = max(max(GAWeight));
GAWeightMaxRowCol=zeros(1,2);
GAWeightMaxRowCol(1,2)= GAWeightCol;
[GAWeightMax,GAWeightRow] = max(max(GAWeight,[],2));
GAWeightMaxRowCol(1,1)= GAWeightRow;

[VyWeightMax,VyWeightCol] = max(max(VyWeight));
VyWeightMaxRowCol=zeros(1,2);
VyWeightMaxRowCol(1,2)= VyWeightCol;
[VyWeightMax,VyWeightRow] = max(max(VyWeight,[],2));
VyWeightMaxRowCol(1,1)= VyWeightRow;

GAWeightMaxRowColCoord = [DeltaGVIMesh(GAWeightRow,1)
XGVICrMesh(1,GAWeightCol)];
VyWeightMaxRowColCoord = [DeltaGVIMesh(VyWeightRow,1)
XGVICrMesh(1,VyWeightCol)];

% Display Various Values
disp(['DeltaGV and XGV for optimized w.r.t. GA, Weights (VGAMax and VGADev):
' num2str(WVGAmax/(WVGAmax+WVGADev)) ' and '
num2str(WVGADev/(WVGAmax+WVGADev))])
disp(GAWeightMaxRowColCoord)

disp(['DeltaGV and XGV for optimized w.r.t. Vy, Weights (VGAMax and VGADev):
' num2str(WVGAmax/(WVGAmax+WVGADev)) ' and '
num2str(WVGADev/(WVGAmax+WVGADev))])
disp(VyWeightMaxRowColCoord)

disp('Max. GV for optimized w.r.t. Vy')
disp(MaxGVIM(VyWeightRow,VyWeightCol))

disp('Max. GA for optimized w.r.t. GA')
disp(MaxGAIM(GAWeightRow,GAWeightCol))
```

```
disp('Max. GV Deviation for optimized w.r.t. Vy')
disp(VyDevNormM(VyWeightRow,VyWeightCol))

disp('Max. GA Deviation for optimized w.r.t. GA')
disp(GADevNormM(GAWeightRow,GAWeightCol))

%Print Gust Curve (GA and GV) at said Point
figure('Name',['GV over Y at optimized point, Weights (VGAMax and VGADev): '
num2str(WVGAmax/(WVGAmax+WVGADev)) ' and '
num2str(WVGADev/(WVGAmax+WVGADev))])
plot(YCoordI2(:,(VyWeightCol-
1)*SmplNoD1New+VyWeightRow),VyMaxIF(:,(VyWeightCol-
1)*SmplNoD1New+VyWeightRow))
figure('Name',['GA over Y at optimized point, Weights (VGAMax and VGADev): '
num2str(WVGAmax/(WVGAmax+WVGADev)) ' and '
num2str(WVGADev/(WVGAmax+WVGADev))])
plot(YCoordI2(:,(GAWeightCol-
1)*SmplNoD1New+GAWeightRow),GAMaxIF(:,(GAWeightCol-
1)*SmplNoD1New+GAWeightRow))

%% Data Collection of Sinngle Design Point

    %% Data Collection

    [fileOverTimeSingle,pathOverTimeSingle]=uigetfile('*.out','Chose the File
to generate single Data Curves:');%chose first File, Folder names need to be
consistent

    % Acces data files (of all data sets)
    OverTimeS=readmatrix([pathOverTimeSingle
fileOverTimeSingle],'FileType','text', 'Delimiter',','
','OutputType','double'); % print data to matrix

    % Search for simulation time of min and max gust angle
    [pksMax,locsMax] =
findpeaks(OverTimeS(:,2),OverTimeS(:,1),'MinPeakDistance',10,'MinPeakProminen
ce',0.01); %find points of max gust angles
    [pksMin,locsMin] = findpeaks(-
OverTimeS(:,2),OverTimeS(:,1),'MinPeakDistance',10,'MinPeakProminence',0.01);
%find points of min gust angles
    pksMin = pksMin*(-1);

    locsMaxLast=locsMax(end); %only take last min and max (where gust
setteld)
    pksMaxLast=pksMax(end);
    locsMinLast=locsMin(end);
    pksMinLast=pksMin(end);

    % figure()
    % plot(locsMinLast,pksMinLast,'o')
    % hold on
    % plot(locsMaxLast,pksMaxLast,'o')
    % plot(OverTime(:,1),OverTime(:,2))

    % Acces data of Vy over y corresponding to above timestep
    fileOverY = 'xy_velocity_tw_topt*-*'; %generate dummy file name
independent of opt1 or opt2 for gust over y according to max and min
    fileOverYArray = dir([pathOverTimeSingle fileOverY]);
    fileOverYArray = {fileOverYArray.name};
```

```
    fileOverYMaxS=fileOverYArray{1,locsMaxLast}; %generate filename of file
corresponding to above timestep
    fileOverYMinS=fileOverYArray{1,locsMinLast};

    OverYMax=readmatrix([pathOverTimeSingle fileOverYMaxS],'FileType','text',
'Delimiter',',','OutputType','double'); %Write data to matrix
    OverYMin=readmatrix([pathOverTimeSingle fileOverYMinS],'FileType','text',
'Delimiter',',','OutputType','double');

    %Y Axis Orientation Normalization (most positive value of Y to most
negative  value of Y
    if OverYMax(1,3) > OverYMax(CellNoY,3)
        OverYMax = OverYMax;
    else
        OverYMax = flip(OverYMax,1);
    end

    if OverYMin(1,3) > OverYMin(CellNoY,3)
        OverYMin = OverYMin;
    else
        OverYMin = flip(OverYMin,1);
    end

    % Collect Data in Matrix over all samples
    VxMaxSingle(:,1) = OverYMax(:,4);
    VxMinSingle(:,1) = OverYMin(:,4);

    VyMaxSingle(:,1) = OverYMax(:,5);
    VyMinSingle(:,1) = OverYMin(:,5);

    GAMaxSingle(:,1) = OverYMax(:,6);
    GAMinSingle(:,1) = OverYMin(:,6);

    %Get Y-Coordinates
    YCoordSingle(:,1) = OverYMax(:,3);



    figure('name','VyMaxSingle')
    plot( YCoordSingle(:,1),VyMaxSingle(:,1))
    hold on
    plot(YCoordI2(:,(VyWeightCol-
1)*SmplNoD1New+VyWeightRow),VyMaxIF(:,(VyWeightCol-
1)*SmplNoD1New+VyWeightRow))
    hold off

    figure('name','VyMinSingle')
    plot( YCoordSingle(:,1),VyMinSingle(:,1))
```

CommaToDot

```
%%Replacing comma decimal seperator with a dot
clc
close all
clear all
%% Variables
FileNo=20; %Number of files to average over


%%Collect Data
[fileZeroAoA,pathZeroAoA]=uigetfile('*.txt','Chose file to start'); %chose
first File, Filename need to be consistent

for c=1:FileNo
    strcounter=append(num2str(c),'.txt');
     if c<10
        fileZeroAoA=fileZeroAoA(1:end-5); %Generate path to different vector
files
        fileZeroAoA=append(fileZeroAoA,strcounter);
     elseif c<100
        fileZeroAoA=fileZeroAoA(1:end-6);
        fileZeroAoA=append(fileZeroAoA,strcounter);
     else
        fileZeroAoA=fileZeroAoA(1:end-7);
        fileZeroAoA=append(fileZeroAoA,strcounter);
    end

    comma2point_overwrite([pathZeroAoA fileZeroAoA]);

end

disp('Done')
%%
    function    comma2point_overwrite( filespec )
    % replaces all occurences of comma (",") with point (".") in a text-file.
    % Note that the file is overwritten, which is the price for high speed.
        file    = memmapfile( filespec, 'writable', true );
        comma   = uint8(',');
        point   = uint8('.');
        file.Data(file.Data==comma) = point;
    end
```

CenterFinder

```
%%Finding the center based on the wake with Gustvanes at zero AoA

%%Made for : 1st column x coordinate, 2nd column y coordinate
%%3rd column x-velocity, 4th column y velocity, first x coordinate smallest
clc
close all
clear all
%% Variables
PoIx=69.9; %Coordinate in flow direction to look in mm
DataWidth=1; %Number of neigbouhring Verctor rows that are avaeraged to
generate data at desired point: eg. 3 --> Data at X=Px and 3 neigbouhring
Vectors at each side
SampelWidthY=325; %Number of total data points in y direction

FileNo=20; %Number of files to average over


%%Collect Data
[fileZeroAoA,pathZeroAoA]=uigetfile('*.txt','Chose file with zero angle of
attack to determin center'); %chose first File, Filename need to be
consistent

ZeroAoA=readmatrix([pathZeroAoA fileZeroAoA],'Delimiter',{'\t'}); % print
initial data to matrix

c=1;
Px=ZeroAoA(c,1);
while Px<PoIx %%Find first Data Point whos X-Coord is bigger then PoIx
    c=c+1;
    Px=ZeroAoA(c,1);
end
PxL=Px;
PxS=ZeroAoA(c-1,1);

PxLDif=abs(PxL-PoIx);
PxSDif=abs(PxS-PoIx);

if PxLDif<PxSDif %%check if first data point with bigger x coordinate or the
corresponding data point with smaller x coordinate is closer to the desired
value
   PoIxR=PxL; %%Chose bigger value
else
    PoIxR=PxS; %%chose smaller value
end

ZeroAoARedAvg=zeros(SampelWidthY,4); %%Genrate Empty Matrix for collection of
reduced and averaged data (only X coordinate of interest)

for c=1:FileNo
    strcounter=append(num2str(c),'.txt');
     if c<10
        fileZeroAoA=fileZeroAoA(1:end-5); %Generate path to different vector
files
        fileZeroAoA=append(fileZeroAoA,strcounter);
     elseif c<100
        fileZeroAoA=fileZeroAoA(1:end-6);
        fileZeroAoA=append(fileZeroAoA,strcounter);
     else
        fileZeroAoA=fileZeroAoA(1:end-7);
```

```
        fileZeroAoA=append(fileZeroAoA,strcounter);
     end

    ZeroAoA=readmatrix([pathZeroAoA fileZeroAoA],'Delimiter',{'\t'}); % print
data to matrix

    ZeroAoARed=zeros(SampelWidthY,4); %%Genrate Empty Matrix for collection
of reduced data (only X coordinate of interest)

    cb=1;
    for ca=1:size(ZeroAoA,1)
        if ZeroAoA(ca,1)==PoIxR
            for cd=0:(DataWidth*2)
            ZeroAoARed(cb,:)=ZeroAoARed(cb,:)+ZeroAoA((ca-DataWidth+cd),:);
            end
            ZeroAoARed(cb,:)= ZeroAoARed(cb,:)/(DataWidth*2+1);
           cb=cb+1;
        end
    end

  ZeroAoARedAvg=ZeroAoARedAvg+ZeroAoARed;
end

ZeroAoARedAvg=ZeroAoARedAvg/FileNo; %%Average over all Samples
ZeroAoARedAvg(:,3)=(-1)*ZeroAoARedAvg(:,3); %%change sign of x velocity (to
positive in flow direction)
ZeroAoARedAvg = flip(ZeroAoARedAvg); %%flip Matrix, so y coordinates increase


[pksWake,locsWake] = findpeaks(-
ZeroAoARedAvg(:,3),ZeroAoARedAvg(:,2),'MinPeakDistance',10,'MinPeakProminence
',1); %search wake
pksWake = pksWake*(-1);

plot(locsWake,pksWake,'o')
hold on
plot(ZeroAoARedAvg(:,2),ZeroAoARedAvg(:,3))

PoIy=(locsWake(1,1)+locsWake(2,1))/2;

c=1;
Py=ZeroAoARedAvg(c,2);
while Py<PoIy %%Find first Data Point whos Y-Coord is bigger then PoIy
    c=c+1;
    Py=ZeroAoARedAvg(c,2);
end
PyL=Py;
PyS=ZeroAoARedAvg(c-1,2);

PyLDif=abs(PyL-PoIy);
PySDif=abs(PyS-PoIy);

if PyLDif<PySDif %%check if first data point with bigger y coordinate or the
corresponding data point with smaller y coordinate is closer to the desired
value
   PoIyR=PyL; %%Chose bigger value
else
    PoIyR=PyS; %%chose smaller value
end
```

```
disp(['The center calculatet with the wake position is at ', num2str(PoIy),'.
This leads two the possible Y-Coordinates of ', num2str(PyL),' and
',num2str(PyS),' of which ',num2str(PoIyR),' is closer.'])
```

PhaseAverage

```
%%Finding the center based on the wake with Gustvanes at zero AoA

%%Made for : 1st column x coordinate, 2nd column y coordinate
%%3rd column x-velocity, 4th column y velocity, first x coordinate smallest
clc
close all
clear all
%% Variables
PoIx=36; %Coordinate in flow direction to look in mm, according to piv (159mm
- -76mm, 159mm is upstream)
DataWidth=2; %Number of neigbouhring Verctor rows that are avaeraged to
generate data at desired point: eg. 3 --> Data at X=Px and 3 neigbouhring
Vectors at each side
SampelWidthY=325; %Number of total data points in y direction
FilePerPhase=5; %Number of files per phase Angle
NoOfAngles=79; %Number of different evaluation points
PhaseAngleSteps=5; %Angle in degree per phase angle
FP=1; %FirsPicture Unusable? 1=Unusable, 0=Usable
vers=2; %Version of Postprocess, used for folder name (v1, v2, vn etc)

%%Collect Data
[fileVecField,pathVecField]=uigetfile('*.txt','Chose file of Series'); %chose
first File, Filename need to be consistent

VecField=readmatrix([pathVecField fileVecField],'Delimiter',{'\t'}); % print
initial data to matrix

mkdir([pathVecField
'PhaseAveragedDataAtx=',num2str(round(PoIx)),'_v',num2str(vers),'\']);

%% Find Data Point of interrest (with regard to x coordinate)
c=1;
Px=VecField(c,1);
while Px<PoIx %%Find first Data Point whos X-Coord is bigger then PoIx
    c=c+1;
    Px=VecField(c,1);
end
PxL=Px;
PxS=VecField(c-1,1);

PxLDif=abs(PxL-PoIx);
PxSDif=abs(PxS-PoIx);

if PxLDif<PxSDif %%check if first data point with bigger x coordinate or the
corresponding data point with smaller x coordinate is closer to the desired
value
   PoIxR=PxL; %%Chose bigger value
else
    PoIxR=PxS; %%chose smaller value
end

%% Phase Average First Angle

VecFieldRedAvg=zeros(SampelWidthY,4); %Generate empty Matrix for collection
of averaged and reduced data

for ca=(1+FP):FilePerPhase %Loop over first Phase Angle
        strcounter=append(num2str(ca),'.txt');
     if ca<10
```

```
        fileVecField=fileVecField(1:end-5); %Generate path to different
vector files
        fileVecField=append(fileVecField,strcounter);
     else
        fileVecField=fileVecField(1:end-6);
        fileVecField=append(fileVecField,strcounter);
     end

    VecField=readmatrix([pathVecField fileVecField],'Delimiter',{'\t'});
%print data to matrix

    VecFieldRed=zeros(SampelWidthY,4); %%Genrate Empty Matrix for collection
of reduced data

    cc=1;
    for cb=1:size(VecField,1) %%Collect Data at X-Coordinate of interest
        if VecField(cb,1)==PoIxR %%Check if Data Point is of interest
            for cd=0:(DataWidth*2) %%Take datapoint and as well as
neigbouhring Data Points
            VecFieldRed(cc,:)=VecFieldRed(cc,:)+VecField((cb-
DataWidth+cd),:);
            end
            VecFieldRed(cc,:)= VecFieldRed(cc,:)/(DataWidth*2+1); %%Average
over neighbouring Datapoints
           cc=cc+1;
        end
    end

  VecFieldRedAvg=VecFieldRedAvg+VecFieldRed; %%Add Up Data over Files at same
Phase Angle
end

VecFieldRedAvg=VecFieldRedAvg/(FilePerPhase-FP); %%Phase Average
VecFieldRedAvg(:,3)=(-1)*VecFieldRedAvg(:,3); %%Change sign of x velocity (to
positive in flow direction)
VecFieldRedAvg = flip(VecFieldRedAvg); %%flip Matrix, so y coordinates
increase

filenamePhaseAveraged=[pathVecField,'PhaseAveragedDataAtx=',num2str(round(PoI
x)),'_v',num2str(vers),'\PhaseAveraged',num2str(0),'deg.txt']; %write phase
averaged data into a .txt file
writematrix(VecFieldRedAvg,filenamePhaseAveraged,'Delimiter','tab');


%% Phase Average all other angles
for c=1:NoOfAngles-1 %Loop over different Phase Angles

VecFieldRedAvg=zeros(SampelWidthY,4); %Generate empty Matrix for collection
of averaged and reduced data

for ca=1:FilePerPhase %Loop over same Phase Angle
    counter=c*FilePerPhase+ca;
    strcounter=append(num2str(counter),'.txt');
     if counter<10
        fileVecField=fileVecField(1:end-5); %Generate path to different sim
results folders
        fileVecField=append(fileVecField,strcounter);
     elseif counter<100
        fileVecField=fileVecField(1:end-6);
        fileVecField=append(fileVecField,strcounter);
     else
```

```
        fileVecField=fileVecField(1:end-7);
        fileVecField=append(fileVecField,strcounter);
     end

    VecField=readmatrix([pathVecField fileVecField],'Delimiter',{'\t'});
%print data to matrix

    VecFieldRed=zeros(SampelWidthY,4); %%Genrate Empty Matrix for collection
of reduced data

    cc=1;
    for cb=1:size(VecField,1) %%Collect Data at X-Coordinate of interest
        if VecField(cb,1)==PoIxR %%Check if Data Point is of interest
            for cd=0:(DataWidth*2) %%Take datapoint and as well as
neigbouhring Data Points
            VecFieldRed(cc,:)=VecFieldRed(cc,:)+VecField((cb-
DataWidth+cd),:);
            end
            VecFieldRed(cc,:)= VecFieldRed(cc,:)/(DataWidth*2+1); %%Average
over neighbouring Datapoints
           cc=cc+1;
        end
    end

  VecFieldRedAvg=VecFieldRedAvg+VecFieldRed; %%Add Up Data over Files at same
Phase Angle
end

VecFieldRedAvg=VecFieldRedAvg/FilePerPhase; %%Phase Average
VecFieldRedAvg(:,3)=(-1)*VecFieldRedAvg(:,3); %%Change sign of x velocity (to
positive in flow direction)
VecFieldRedAvg = flip(VecFieldRedAvg); %%flip Matrix, so y coordinates
increase

filenamePhaseAveraged=[pathVecField,'PhaseAveragedDataAtx=',num2str(round(PoI
x)),'_v',num2str(vers),'\PhaseAveraged',num2str(c*PhaseAngleSteps),'deg.txt']
; %write avaraged data to .txt file

writematrix(VecFieldRedAvg,filenamePhaseAveraged,'Delimiter','tab');


end
```

Calculations

```
%%Finding the center based on the wake with Gustvanes at zero AoA

%%Made for : 1st column x coordinate, 2nd column y coordinate
%%3rd column x-velocity, 4th column y velocity, first x coordinate smallest
clc
close all
clear all
%% Variables
PoIy=0.6503; %Y coordinate which represents reallife zero (from center
finder)
SampelWidthY=325; %Number of total data points in y direction
NoOfFiles=79; %Number of Phaseaveraged Files
DegStep=5; %Degree Step
FilterWinY=30;
FilterWinT=20;
AverageWidthY=2; %No. of neighbouring cells in y direction which are used to
average, moving average, if 0 no, no averaging is performed in y direction


%% Collect Data
[filePA,pathPA]=uigetfile('*.txt','Chose first Phase Averaged File'); %chose
first File, Filename need to be consistent

PAVec=readmatrix([pathPA filePA],'Delimiter',{'\t'}); % print initial data to
matrix

c=1;
Py=PAVec(c,2);
while Py<PoIy %%Find first Data Point whos Y-Coord is bigger then PoIY
    c=c+1;
    Py=PAVec(c,2);
end
PyL=Py;
PyS=PAVec(c-1,2);

PyLDif=abs(PyL-PoIy);
PySDif=abs(PyS-PoIy);

if PyLDif<PySDif %%check if first data point with bigger Y coordinate or the
corresponding data point with smaller Y coordinate is closer to the desired
value
   PoIyR=PyL; %%Chose bigger value
   loc=c;
else
   PoIyR=PyS; %%chose smaller value
   loc=c-1;
end

%% Collect Time dependet Data At (reallife) Y=0

TimeDat=zeros(NoOfFiles,4); %%Genrate Empty Matrix for collection of time
dependent data at y = PoIyR

for c=0:NoOfFiles-1

    strcounter=append(num2str(c*DegStep),'deg.txt');
    filePA=filePA(1:13); % Generate path to different Vectordata Files
    filePA=append(filePA,strcounter);
```

```
    PAVec=readmatrix([pathPA filePA],'Delimiter',{'\t'}); % print data to
matrix

    TimeDat(c+1,1)=c*DegStep;    %Phase Angle

    %TimeDat(c+1,2)=PAVec(loc,3); %X-Velocity No averaging

    %TimeDat(c+1,3)=PAVec(loc,4); %Y-Velocity No averaging

    %TimeDat(c+1,4)=atan(PAVec(loc,4)/PAVec(loc,3)); %GustAngle in Radians No
Averaging

    AverageYdir=zeros(1,2); %averaging in y direction, first column vx,
second vy
    for ca=0:(AverageWidthY*2) %%Take datapoint and as well as neigbouhring
Data Points
        AverageYdir(1,1)=AverageYdir(1,1)+PAVec((loc-AverageWidthY+ca),3);
%Vx
        AverageYdir(1,2)=AverageYdir(1,2)+PAVec((loc-AverageWidthY+ca),4);
%Vy
    end
    AverageYdir= AverageYdir/(AverageWidthY*2+1); %%Average over neighbouring
Datapoints
    TimeDat(c+1,2) = AverageYdir(1,1);
    TimeDat(c+1,3) = AverageYdir(1,2);
    TimeDat(c+1,4) = atan(AverageYdir(1,2)/AverageYdir(1,1));

end
[pks,locs] = findpeaks(-
TimeDat(:,3),TimeDat(:,1),'MinPeakDistance',300,'MinPeakProminence',0.01);%fi
nd points of min gust angles
pks = pks*(-1);
locs;

TimeDatS=smoothdata(TimeDat,1,'sgolay',FilterWinT);

TimeDatComplete=[TimeDat zeros(NoOfFiles,1) TimeDatS];

fileNameTimeData = [pathPA,'DataOverTime.xlsx']; %write avaraged data to .txt
file
writematrix(TimeDatComplete,fileNameTimeData);

figure()
plot(locs,pks,'o')
hold on
plot(TimeDat(:,1),TimeDat(:,3)) %Vy over time
hold on
plot(TimeDat(:,1),TimeDatS(:,3)) %Vy smoothend over time
hold on
plot(TimeDat(:,1),(TimeDat(:,4)/(2*pi)*360)) %Ga over time (unsmoothend

%% Collect Y dependet Data At max Amplitude

fileMaxVy=append(filePA(1:13),append(num2str(locs),'deg.txt'));
PAMaxVy=readmatrix([pathPA fileMaxVy],'Delimiter',{'\t'}); % print data to
matrix
PAMaxVyAvg=movmean(PAMaxVy,(AverageWidthY*2+1),1); %Averaging Data in y
directin with sliding average
```

```
PAMaxVyS=smoothdata(PAMaxVyAvg,1,'sgolay',FilterWinY);

PAMaxComplete = [PAMaxVyAvg zeros(SampelWidthY,1) PAMaxVyS]; % add smoothend
data to original data

fileNameYData = [pathPA,'DataOverY_Sample=',num2str(locs),'deg.xlsx']; %write
avaraged data to .txt file
writematrix(PAMaxComplete,fileNameYData);

figure()
plot(PAMaxVyAvg(:,2),PAMaxVyAvg(:,4)) %Vy
hold on
plot(PAMaxVyAvg(:,2),PAMaxVyAvg(:,3)/30) %Vx Scaled to keep Vy visible
hold on
plot(PAMaxVyAvg(:,2),(PAMaxVyAvg(:,4)./PAMaxVyAvg(:,3))/(2*pi)*360) %Ga
hold on
plot(PAMaxVyS(:,2),PAMaxVyS(:,4)) %Vy smoothend
```

Appendix C

UDF_1-cos

```
#include "udf.h"
DEFINE_CG_MOTION(OneMinCos, dt, vel, omega, time, dtime)
{
        NV_S (omega, =, 0.0);
        if (time <= 0.05)
                omega[2] = 0.0;
        else if (0.05 < time && time <=0.1333333 )
                omega[2] = (M_PI / 3) * 2 * M_PI * cos((2 * M_PI * 12 * (time - 0.05)) - (M_PI / 2));
        else
                omega[2] = 0.0;
}
```

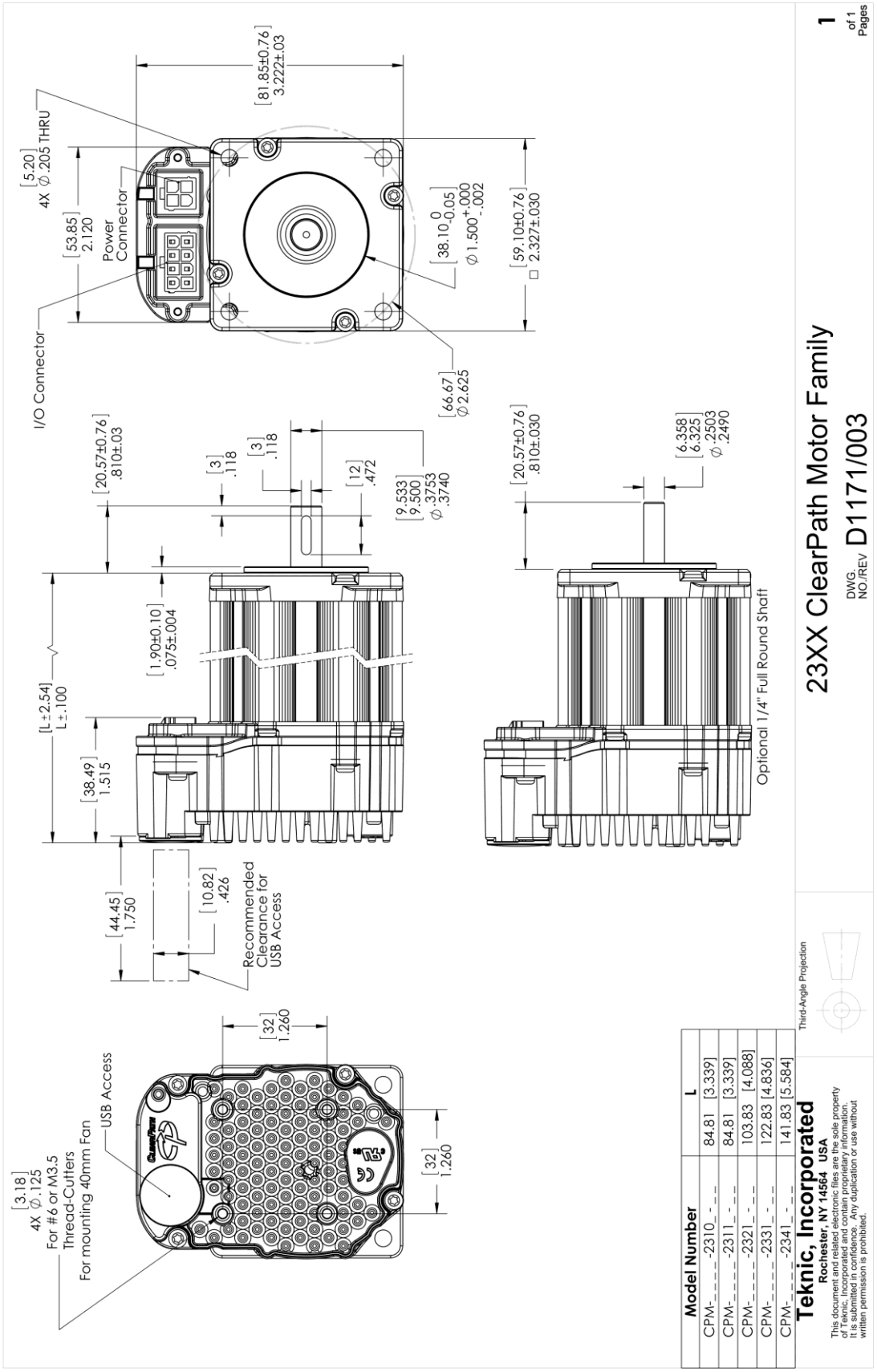
UDF_sin

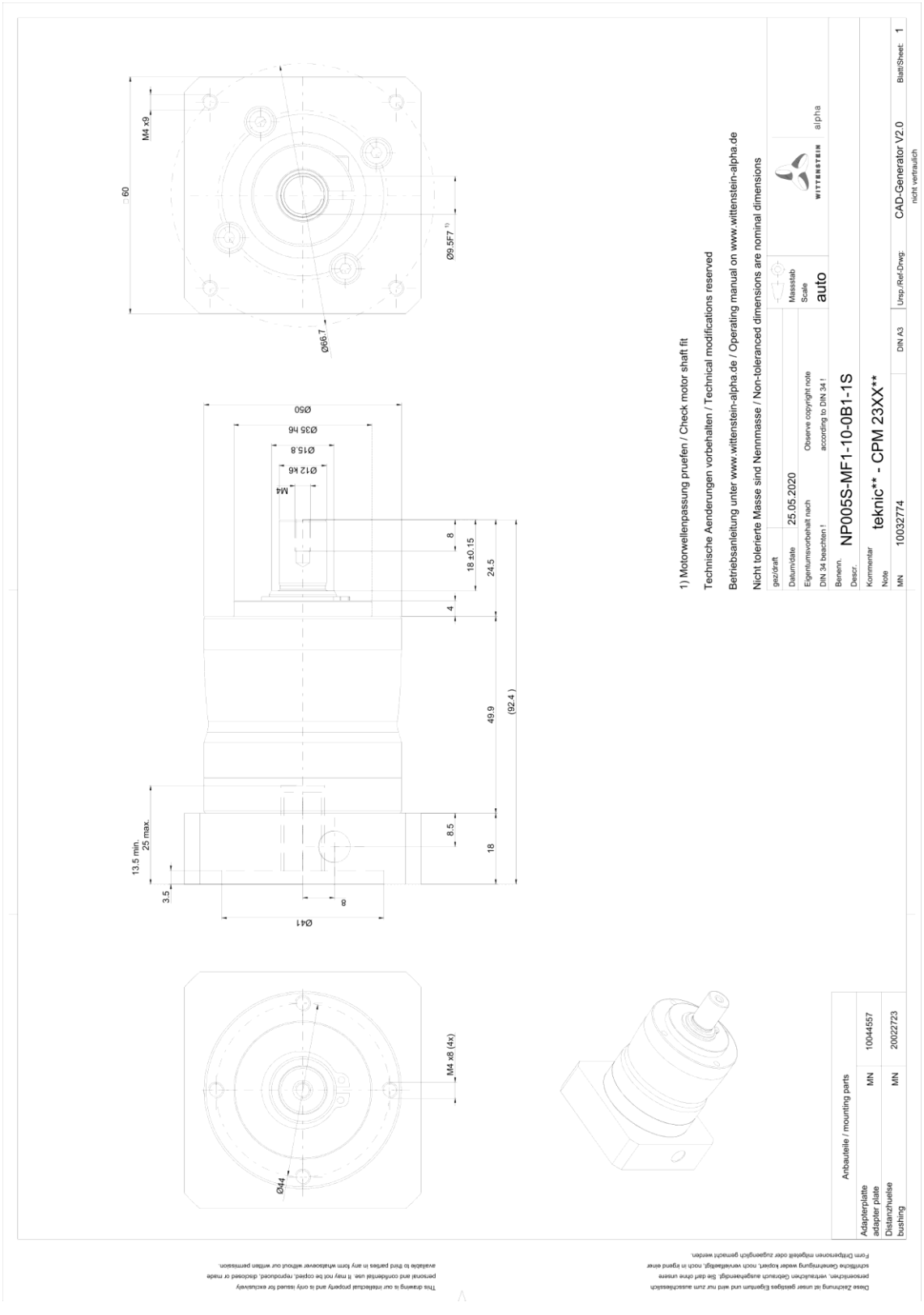
```
#include "udf.h"
DEFINE_CG_MOTION(GVU, dt, vel, omega, time, dtime)
{
        real freq_param, SdyTime, COG_JumpX, COG_JumpY;
        freq_param = RP_Get_Input_Parameter("GustVane_Freq");
        COG_JumpX = RP_Get_Input_Parameter("GV_TransX");
        COG_JumpY = RP_Get_Input_Parameter("GV_TransY");
        SdyTime = 3 / (5 * freq_param);
        NV_S (vel, =, 0.0);
        NV_S (omega, =, 0.0);

        if (time < (1 / (freq_param * 119)))
                vel[0] = COG_JumpX/(1 / (freq_param * 119));
```

```
        else
                vel[0] = 0;
        if (time < (1 / (freq_param * 119)))
                vel[1] = COG_JumpY/(1 / (freq_param * 119));
        else
                vel[1] = 0;
        if (time <= SdyTime)
                omega[2] = 0.0;
        else
                omega[2] = (M_PI / 18) * 2 * M_PI * freq_param * cos(2 * M_PI * freq_param * (time -
SdyTime));
}
DEFINE_CG_MOTION(GVL, dt, vel, omega, time, dtime)
{
        real freq_param, SdyTime, COG_JumpX, COG_JumpY;
        freq_param = RP_Get_Input_Parameter("GustVane_Freq");
        COG_JumpX = RP_Get_Input_Parameter("GV_TransX");
        COG_JumpY = RP_Get_Input_Parameter("GV_TransY");
        SdyTime = 3 / (5 * freq_param);
        NV_S(vel, =, 0.0);
        NV_S(omega, =, 0.0);

        if (time < (1 / (freq_param * 119)))
                vel[0] = COG_JumpX / (1 / (freq_param * 119));
        else
                vel[0] = 0;

        if (time < (1 / (freq_param * 119)))
```

```
                vel[1] = -1 * (COG_JumpY / (1 / (freq_param * 119)));
        else
                vel[1] = 0;


        if (time <= SdyTime)
                omega[2] = 0.0;
        else
                omega[2] = (M_PI / 18) * 2 * M_PI * freq_param * cos(2 * M_PI * freq_param * (time -
SdyTime));
}
```

Appendix D


23XX ClearPath Motor Family
DWG. NO./REV D1171/003
1
of 1 Pages
Third-Angle Projection
Teknic, Incorporated
Rochester, NY 14564 USA
Model Number
L
CPM-_ _ _ _-2310_-_ _ _ 84.81 [3.339]
CPM-_ _ _ _-2311_-_ _ _ 84.81 [3.339]
CPM-_ _ _ _-2321_-_ _ _ 103.83 [4.088]
CPM-_ _ _ _-2331_-_ _ _ 122.83 [4.836]
CPM-_ _ _ _-2341_-_ _ _ 141.83 [5.584]
I/O Connector
Power Connector
[5.20] 4X Ø.205 THRU
[53.85] 2.120
[81.85±0.76] 3.222±.03
[38.10 0 -0.05] Ø1.500 +.000 -.002
[59.10±0.76] □ 2.327±.030
[66.67] Ø2.625
[20.57±0.76] .810±.03
[3] .118
[12] .472
[9.533] [9.500] Ø.3753 .3740
[1.90±0.10] .075±.004
[L±2.54] L±.100
[38.49] 1.515
[44.45] 1.750
[10.82] .426
Recommended Clearance for USB Access
[6.358] [6.325] Ø.2503 .2490
Optional 1/4" Full Round Shaft
[3.18] 4X Ø.125 For #6 or M3.5 Thread-Cutters For mounting 40mm Fan
USB Access
[32] 1.260
[32] 1.260



WITTENSTEIN | alpha

	Customer data	Your contact
Company Contact Street ZIP / Town Phone E-mail		WITTENSTEIN Christophe Baert 9270 christophe.baert@wittenstein.biz
Your project:	**Project 9**	Date: 25.05.2020
Your variation:	**Variant 1**	
Your axes:	**As 1**	

1 General notes

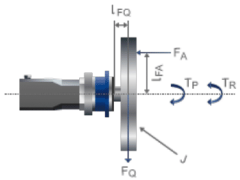
The data provided by you forms the basis for the calculation performed with cymex®. WITTENSTEIN alpha does not assume any responsibility for the accuracy and completeness of this data – by ordering, you are confirming the accuracy of the data provided by you.
The calculation documentation is based on your customized design. This non-binding recommendation does not release you from your duties or responsibilities, which include compliance with legal and safety regulations, and completion of a functional test. We remind you that the permitted load values for the selected products may not be exceeded under any operating conditions.
This recommendation, which is based on the calculation performed with cymex® is not suitable for application and/or transfer to other products, data, applications, etc.
Please note that the cymex® calculation documentation is protected by copyright.

2 Your application

2.1 Project description

2.2 Rotary load application

Parameter	Value	Parameter	Value
Axial force F_A	0 N	Inertia J	0.000532 kgm²
Lateral force F_Q	0 N	Max. process torque T_P	2 Nm
Lever arm axial force l_{FA}	0 mm	Max. angular speed n_{Max}	105.75 min⁻¹
Lever arm lateral force l_{FQ}	0 mm	Max. angular acceleration α_{Max}	836.27 rad/s²
Friction torque T_R	0 Nm	Inertia for lambda calculation J_{Lambda}	0.000532 kgm²

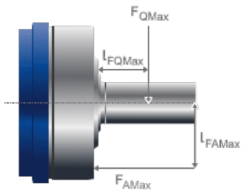


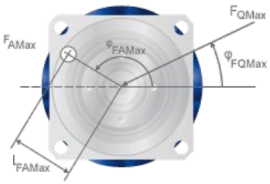
Missing data has been replaced with empirical data, which must be checked and corrected if necessary.

2.3 Manual force for flange

Parameter	Value
Axial force F_{AMax}	0 N
Lever arm axial force l_{FAMax}	0 mm
Angle axial force φ_{FAMax}	0 °
Lateral force F_{QMax}	0 N
Lever arm lateral force l_{FQMax}	0 mm
Angle lateral foce φ_{FQMax}	0 °

WITTENSTEIN | alpha



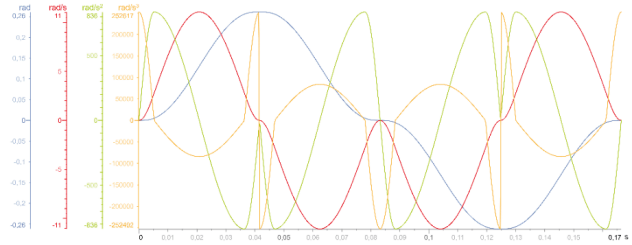


Attention: These parameters have been modified afterwards and substitute the respective values mentioned above.

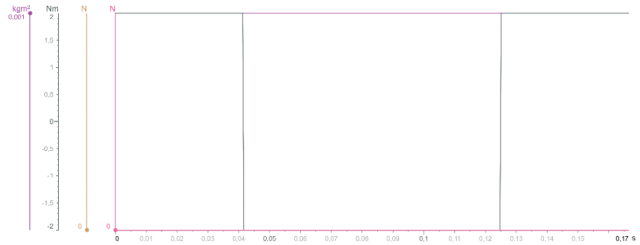
2.4 Cycle data

Parameter	Value	Parameter	Value
Movement duration t_{mov}	0.17 s	Duty cycle ED	100 %
Cycle duration t_{cyc}	0.17 s	Number of cycles per hour n/h	86538.46

2.5 Movement profile



2.6 Load profile



3 Drive design

3.1 Gearbox

Type: **Low backlash planetary gearbox NP**

Designation: **NP005S-MF1-10-0B1-1S**

Utilization total: 34 %

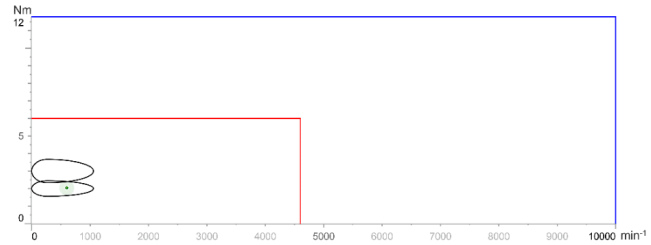
Features	Value
Ratio i	10
Output design	Smooth shaft
Max. Backlash value j_t	≤ 10 arcmin
Design keyword	Standard

WITTENSTEIN | alpha

Parameter	Calculated Value	Permissible Value	Utilization
Max. acceleration torque at output (with impact factor 1.5) $T_{2\alpha}$ (fs)	3.66 Nm	11.79 Nm	31 %
Nominal torque at output T_{2N}	2.04 Nm	6 Nm	34 %
Max. speed rotative n_{1Max}	1057.45 min^{-1}	10000 min^{-1}	11 %
Nominal speed rotative n_{1N}	600.94 min^{-1}	4600 min^{-1}	13 %
Max. axial force F_{2AMax}	0 N	700 N	0 %
Max. radial force F_{2RMax}	0 N	800 N	0 %
Max. tilting moment M_{2KMax}	0 Nm	23 Nm	0 %
Ratio of inertia lambda λ with respect to gearbox output	0.38	-	-
Safety shaft against flow S_F	26.5	-	-
Safety shaft against fatigue fracture S_D	17.26	-	-
Bearing lifetime L_{h10}	>20000 h	-	-

*) WITTENSTEIN recommends using this λ value for determining the control precision

Gearbox characteristic curve



Gearbox characteristic curve key	
Grey	Characteristic curve application
Black	Characteristic curve application (with load factor 1.5)
Green	Operating point of application (T_N and n_{1N})
Red	S1 characteristic curve
Blue	S5 characteristic curve

Messages

Errors

Warnings

Notes

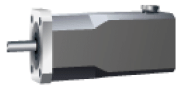
WITTENSTEIN | alpha

3.2 Motor

Manufacturer: teknic**

Type: CPM 23XX**

Utilization total: 103 %

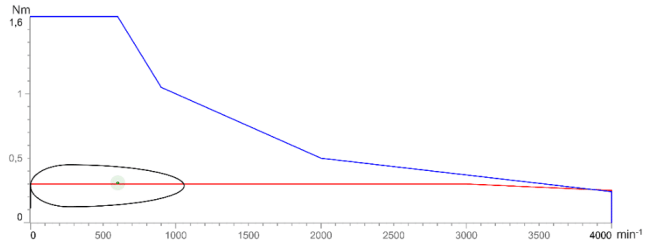


Features	Value
Inertia J	0.1 kgcm²
Shaft diameter d	9.5 mm

Parameter	Calculated value	Permissible value	Utilization
Max. torque T_{Max}	0.45 Nm	1.6 Nm	28 %
Max. speed n_{Max}	1057.45 min^{-1}	4000 min^{-1}	26 %
Nominal speed n_N	600.94 min^{-1}	3000 min^{-1}	20 %
Ratio of inertia lambda λ with respect to motor shaft	0.93	-	-
Average torque T_{RMS}	0.31 Nm	See S1 characteristic curve	-
Utilization curve (S1)	-	-	103 %
Max. utilization curve (S5)	-	-	37 %

The motor in the application design is included for example purposes only. It is the responsibility of the manufacturer to confirm the suitability of the motor. Please note that some values may not be defined by certain manufacturers.

Motor characteristic curves



Motor characteristic curve key	
Black	Characteristic curve application
Green	Operating point application (T_{RMS} und n_N)
Red	S1 characteristic curve
Blue	S5 characteristic curve