University Carlos III of Madrid

Bachelor Thesis

AEROSPACE ENGINEERING

DEVELOPMENT OF AN AERODYNAMIC MODEL FOR A FLEXIBLE KITE FOR WIND POWER GENERATION

Author: Roberto Peña Cano

Tutors: Óscar Flores Arias &

Felipe Hernández San Felipe

Acknowledgements

Firstly, I would like to express my sincere gratitude to my thesis supervisor, Felipe Hernández, and also to my thesis director, Óscar Flores, for the continuous support and research, for their patience, enthusiasm and immense knowledge with which they guided me through the development of this thesis, for which I will always be grateful.

I would also like to thank all the people who helped me during the past four years, specially my classmates and friends, with whom I have shared many good times, and without them this stage of my life had never been the same.

Last but not least, I would like to the thank my parents and my brothers for their unconditional support, for being always with me either in good or adverse moments during my entire life.

Abstract

In this bachelor's thesis an Aerodynamic an Structural model have been developed for the study of a flexible kite. With this model, a designer is able to obtain a preliminary view of the forces acting on a kite as well as the final shape of the kite after the equilibrium of forces is reached.

The project can be clearly divided in two parts: The Aerodynamic model and the Structural model.

The Aerodynamic model is a 3D Vortex lattice method. The Biot-Savart law was used to get the flow-field around a finite straight vortex line. With the boundary condition that the air cannot flow through the kite, the strength of the flow-field induced by the vortices can be computed. Also the Kutta-Joukowski theorem is used to calculate the force produced by each vortex segment. Once the forces are known, the lift and the induced drag are also computed. This forces will be used later, together with the structural model, in order to obtain the equilibrium position of the kite.

The Structural model is obtained applying two conditions: each segment that compose the kite should be in equilibrium, i.e. the forces and the moments acting on each segment should be equal to zero. The other condition is that the two tethers come together in one point. It is indispensable that all the segments in which the kite is divided have the same length, to ensure the convergence. All of these condition will be explain better in section 3. A remarkable assumption is that the stiffness of the kite is not taken into account.

Both the Aerodynamic model and the Structural one have been developed using Matlab ®.

Contents

Acknowledgements **I**

Abstract **II**

List of Figures **VI**

List of Tables **VIII**

Glossary **IX**

1 Introduction **1**

1.1 The Pumping Cycle Principle 3

1.2 Different types of kites 4

1.2.1 Dimensions of a kite 7

1.3 Socio-economic Aspect 8

1.4 Aim of the present work 9

1.5 Legislation 9

2 Vortex Lattice Method **10**

2.1 Introduction 10

2.2 Physical problem 13

2.3 Mathematical analysis 14

2.3.1 Potential flow 14

2.3.2 Boundary conditions 16

2.3.3 Biot-Savart law 17

2.3.4 Horseshoe vortex 18

2.3.5 Selection of Control Point/Vortex location 20

2.4 Computational Process Without Wake Vortex Interaction 20

2.4.1 Geometry and Flow Data 20

2.4.2 Discretization and Grid Generation 20
2.4.3 Routine Functions 22
2.4.4 Influence Coefficients 25
2.4.5 Establish RHS Vector and Solve Linear Set of Equations 27
2.4.6 Secondary Computations 27
2.5 Validation of the Aerodynamic Code 28
2.5.1 Tornado Code 29
2.5.2 XFLR5 31

3 Structural model **34**
3.1 Nomenclature 34
3.2 Geometric constraint / Compatibility Equations 35
3.3 Steady state 36
3.3.1 Calculation of Reactions and Tensions $T_1(\overline{\theta})$ and $T_{N+1}(\overline{\theta})$: 37
3.3.2 Calculation of Tensions in each panel 40
3.3.3 Calculation of Moments in each panel 41

4 Computational method **43**
4.1 Input parameters 43
4.2 Newton-Raphson Method [3] 43
4.2.1 Relaxation Parameter 47
4.3 Errors 47
4.4 Flowchart 48
4.4.1 Jacobian 48

5 Results **50**
5.1 Kite geometry and forces 51
5.2 Aerodynamic coefficients 52
5.3 Kite parameters 53
5.4 Final comparison 54

6 Conclusions and Future work **56**
6.1 Conclusions 56
6.2 Future work 57

Appendices **59**

A — Appendix: Selection of Control Point/Vortex Location **60**
A.0.1 Simplest Approach: A Flat Plate 60
A.0.2 Determine placement of the vortex using parabolic camber model . 62

B — Appendix: Budget **65**

C — Appendix: Codes **66**

References **80**

List of Figures

1 Schematic representation of the Energy scenario . . . 3
2 Schematic representation of the pumping cycle's traction and retraction phases . . . 4
3 Ram air traction kites . . . 5
4 Modern ram air traction kite . . . 5
5 Leading Edge Inflatable kite . . . 6
6 BOW kite . . . 7
7 Typical split of functions in a CFD software system . . . 10
8 Steps in applying computational analysis to aerodynamics . . . 11
9 Analogy between computational and experimental aerodynamics . . . 13
10 Simplified physical models and corresponding mathematical equations . . . 14
11 Decomposition of a general airfoil at certain incidence . . . 16
12 Point Vortex Flow . . . 17
13 Vortex filament . . . 18
14 Horseshoe Vortex Element . . . 19
15 The horseshoe vortex lattice model for the classical vortex lattice method [13] . . . 21
16 A spanwise horseshoe vortex element [13] . . . 22
17 Planforms . . . 30
18 Comparison of the theoretical and the experimental lift coefficients for the swept wing of figure 17 in a subsonic stream . . . 31
19 Planforms . . . 32
20 Comparison of the theoretical and the experimental lift and induced drag coefficients for the kite of figure 19 in a subsonic stream . . . 33
21 Nomenclature . . . 34
22 Forces and Tensions . . . 36
23 Tensions . . . 38

24 Forces acting on each panel 40
25 Moments in each panel 41
26 Solution of two nonlinear equations in two unknowns. Solid curves refer to $f(x, y)$, dashed curves to $g(x, y)$. Each equation divides the (x, y) plane into positive and negative regions, bounded by zero curves. The desired solutions are the intersections of these unrelated zero curves. The number of solutions is a priori unknown [17] 45
27 Flowchart 49
28 Geometry of the kite 51
29 Geometry of the kite for three angles of attack 52
30 Aerodynamic coefficients 53
31 Kite parameters 54
32 Lift coefficient vs Angle of attack 55
33 The notation for control point and vortex location analysis [4] 60

List of Tables

2 AR Ranges [2] 7

3 Input Parameters 44

4 Input Parameters for the practical case 50

5 Project Budget 65

Glossary

Symbol	Unit	Definition
Abbreviations		
LEI	$[-]$	Leading Edge Inflatable Kite
VLM	$[-]$	Vortex Lattice Method
KCU	$[-]$	Kite Control Unit
RHS	$[-]$	Right Hand Side
Greek symbols		
α	$[rad]$	Angle of attack
ω	$[1/s]$	Vorticity
ρ	$[kg/m^3]$	Air density
∇	$[-]$	Gradient operator
$\nabla\times$	$[-]$	Curl operator
$\nabla\cdot$	$[-]$	Divergence operator
∇^2	$[-]$	Laplace operator
ϕ	$[-]$	Velocity potential
Φ	$[-]$	Velocity potential function
Γ	$[m^2/s]$	Circulation
θ	$[rad]$	Kite angles
Latin symbols		
b	$[m]$	Kite span
c	$[m]$	Kite chord
S	$[m^2]$	Kite area
L_0	$[m]$	Tether length
g	$[m/s^2]$	Gravitational acceleration

V	$[m/s]$	Wind speed
C_L	$[-]$	Lift coefficient
$C_{D_{ind}}$	$[-]$	Induced drag coefficient
L	$[N]$	Lift
D	$[N]$	Drag
J	$[-]$	Jacobian
R	$[N]$	Reactions
T	$[N]$	Tensions
Q	$[m/s]$	Free stream velocity
w	$[m/s]$	Normal velocity component
V_θ	$[m/s]$	Tangential velocity
r	$[m]$	Radius
$\overrightarrow{n}$	$[-]$	Normal vector
C_p	$[-]$	Pressure coefficient

Subscripts

∞	$[-]$	At infinity
$upper$	$[-]$	Upper surface
$lower$	$[-]$	Lower surface
θ	$[-]$	Tangential component

1 Introduction

The issues postured by electric energy generation from fossil sources incorporate high expenses because of huge demand and restricted resources, contamination and CO_2 generation, and the geopolitics of producer nations. These issues can be overcome by alternative sources that are renewable, cheap, sustainable and easily available. However, current renewable technologies have limitations. For sure, even the most idealistic conjecture on the diffusion of wind, photovoltaic, and biomass sources assesses close to a 20% contribution to total energy production within the following 15-20 years.

The answer to the growing energy needs of the world could be in the wind. This renewable energy can be exploited in a more effective way than the wind turbines do it nowadays, using only the currents of air closest to the earth since they often have an average height of 80 meters and rarely reach 150. Except very windy places, usually, the wind speed at 80 meters above the ground is 4.6 meters per second. At 800 meters, however, the air circulates much faster, reaching 7.2 meters per second on average [8], a figure that offers exceptional possibilities for the production of electricity.

The challenge will be to reach this height with a device that survives at that altitude and can take advantage of the wind. The solution might be in the hands of the children: the kites. An evolution of this toy, the sail that the lovers of the "Sky Surf" use to fly over the waves, is the key of a revolutionary system that a group of Italian investigators is developing. As they say, they can produce thanks to the wind, so much energy as an atomic power plant. The idea arose thanks to the meeting between a lover of the "Sky Surf" and owner of a company of automated systems, Massimo Ippolito, and a teacher of automatic controls of the Technical University of Turin, Mario Milanese. Ippolito, Milanese and the third partner have constituted Kite Gen, whose aim is "to offer efficient solutions to the shortage of planetary energy" by means of renewable energies "cheap, abundant and not pollutants ".

In order that Kite Gen kite's take advantage of the energy of the wind, the investigators have planned a system that have baptised "the yo-yo". It consists of holding for with two cables of 800 meters to the flying device so that, as it ascends, makes turn two cylinders that, like dynamo, produce energy. When the kite has reached his maximum height, an engine gathers the cables and the flight returns to the beginning. For it, only it spends 15% of the electricity produced before. Kite Gen's driving people have designed in addition a system of navigation that, by means of sensors adhered to the kite, gives information to the cylinders of the base so that the kite is flying always in a figure-eight pattern. This way, the utilization of the wind and the energetic production are maximized.

By causing this movement, Kite Gen's kites behave as if they were the exterior part of the propeller of a turbine, though placed in the place where the wind is stronger. The plants of electrical production planned by Kite Gen would be formed by several of these kites with their respective bases and a center of control from which the flight of the kites is piloted according to the information that offer the sensors. In agreement to their projections, the kites might manage to produce up to 1.000 megawatts of energy if 200 of them were driving a ring, like carousel. This type of power plant, which would generate the same energy that a nuclear medium plant, it would cost between 500 and 600 million Euros, a sixth part of the price of the atomic one. The produced electricity would be also very cheap, since, according to Kite Gen's calculations, the operation costs would represent a third of those of the cheapest and most used energy source, the coal (see Figure 1).

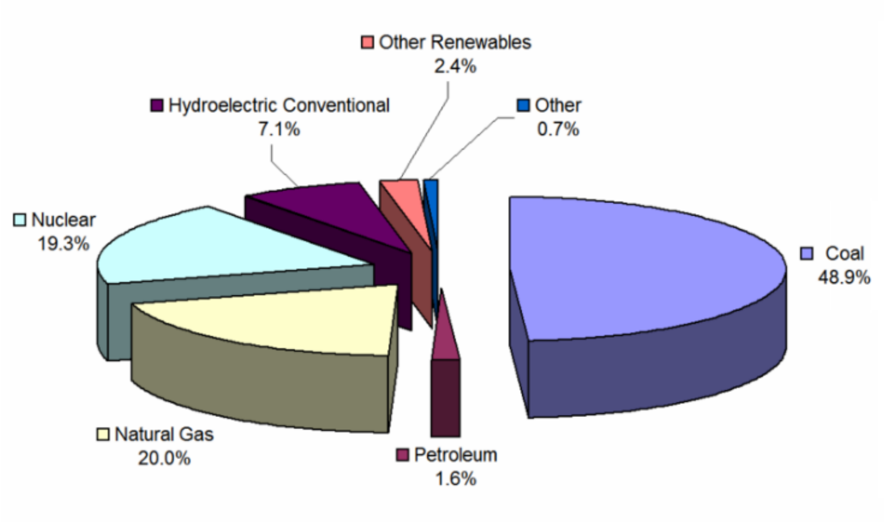


Figure 1: Schematic representation of the Energy scenario

1.1 The Pumping Cycle Principle

In order to produce energy a pumping cycle has to be used. This pumping cycle system consists of a ground station, a tethered kite and an additional kite control unit (KCU). The ground station contains a winch and a generator/motor. The tethered kite varies in design, shape and construction material depending on the research group.

The pumping cycle consists of two phases: traction and retraction. The traction phase is the productive part of the cycle where energy is extracted from the wind. On the contrary, in the retraction phase the energy is consumed while the winch reels the kite back in. These phases are illustrated in Figure 2.

During the traction phase, the kite is flown at high angles of attack in a figure-eight pattern. The way to produce energy is converting to electrical power the high traction forces by turning a winch connected to a generator while reeling out the kite. However, during the retraction phase the kite's angle of attack is reduced in order to minimize the traction force.

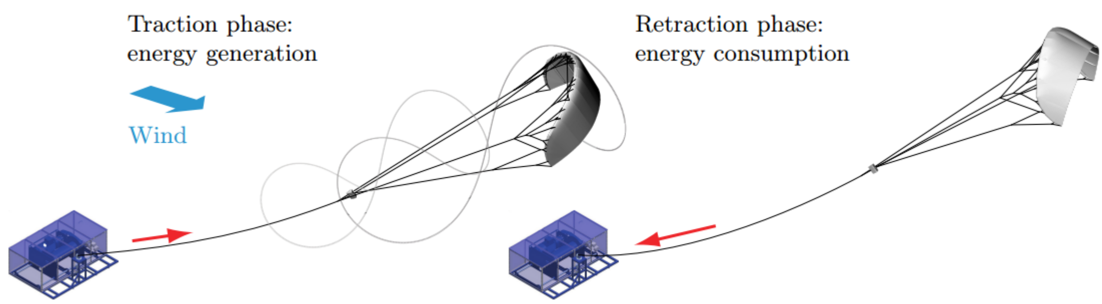


Figure 2: Schematic representation of the pumping cycle's traction and retraction phases

In the end, the consumed energy is less than the produced energy resulting in net energy production. The system efficiency with respect to the kite can be increased by having higher lift-to-drag ratio and/or by increasing the de-power range. This de-power range of the kite is the difference between the traction and retraction force.

1.2 Different types of kites

A briefly introduction to the most relevant types of kites is going to be presented in this subsection.

Ram air traction kites

The evolution of the original Flexifoil® design is still flying and is greatly appreciated for the huge amount of pull that it can generates. This type of kite offers the interesting possibility to add more and more kites to a single couple of control lines, as it can be seen in Figure 3.

Figure 3: Ram air traction kites

A modern ram air traction kite is shown in Figure 4. This kind of kites need a complex bridle system to maintain the shape. The control lines has been progressively increased up to 5. The additional control lines give the possibility to control the kite angle of attack in a easier way.

Figure 4: Modern ram air traction kite

Leading Edge Inflatable kite

This kind of kites has a series of inflatable bladders. That means that an extremely light wing can support a certain amount of structural stiffness limiting the deformations under the flight loads, that greatly affect the kite shape. However, the flow in the boundary layer can easily detaches from the wing lower surface due to unconventional wing section. The designers trade-off is currently trying to reduce to a minimum the leading edge structure diameter. A sample of LEI kite is shown in Figure 5. [14]

Figure 5: Leading Edge Inflatable kite

BOW kites

The better way to describe this type of kite is using directly the inventors explanation: *the BOW concept consists in giving some sweep to the kite wing to flatten its trailing edge then flattening the leading edge in a mechanical way to match the shape of the trailing edge and thus obtaining a flatter wing with large depower.* The term "depower" refers to the pilot capability to suddenly reduce the lift maintaining the kite control when dangerous conditions arise (gusts, obstacles, manoeuvring errors). A sample of BOW kite is shown in Figure 6.

Figure 6: BOW kite

1.2.1 Dimensions of a kite

Since the kites studied have rectangular shape, one of the most important parameter when designing a kite is the Aspect Ratio. This is the most visible parameter that the user will see since the Aspect Ratio determines the shape of the kite.

Higher Aspect Ratio kites have less induced drag (upwash and tip vortex effects) than Lower AR kites of the same characteristics. Induced drag is inverse proportional to AR (Equation 2.5.1). So when stationary at the wind window, a low AR kite can generate the same amount of pull as a higher AR kite (of the same characteristics) but as soon as we need to move the kite for more power, a higher AR kite can accelerate faster therefore get more power sooner than a low AR kite.

As a rule of thumb, a higher AR kite has a larger Power Window than a lower AR kite. The Power window is the difference between min power and max power that can be obtained from the kite.

Following are the recommended AR ranges:

Kite Type	Very Low AR	Low AR	Moderate AR	High AR	Very High AR
Foil	2.5	3	4	5	5.5
Inflatable / Arc	3	4	5	6	7

Table 2: AR Ranges [2]

Also the chord is another important parameter in the design of the kite. This parameter has a range between 0.5 and 3 meters depending on the AR chosen. The characteristic area of commercial kites is up to 10 m^2.

1.3 Socio-economic Aspect

At present, a small scale yo-yo prototype has been accomplished. This system can generate up to 40 kW using commercial kites with characteristic area up to 10 m^2 and line length up to 800 m. The prototype is under test. Preliminary tests show that the amount of energy predicted by simulation is confirmed by experimental data.

A new KiteGen prototype is expected to be built to demonstrate the energy generation capabilities of the carousel configuration. In particular, a carousel structure with a single kite steering unit mounted on a cart riding on a circular rail will be considered. To collect the energy produced by the wagon motion, the wheels of the cart are connected to an alternator. Such a prototype is expected to produce about 0.5 MW with a rail radius of about 300 m. According to scalability, a platoon of carts, each one equipped with a kite steering unit, can be mounted on the rail to obtain a more effective wind power plant. This configuration can generate, on the basis of preliminary computations, about 100 MW at a production cost of about 20 €/MWh, which is two to three times lower than from fossil sources.

The carousel configuration is scalable up to several hundred megawatts, leading to increasing advantages over current wind farms. Using data from the Danish Wind Industry Association Web site [1], it follows that, for a site such as Brindisi, in the south of Italy, a 2-MW wind turbine has a mean production of 4000 MWh/year. To attain a mean generation of 9 TWh/year, which corresponds to almost 1000-MW mean power, 2250 such towers are required, with a land usage of 300 km^2 and an energy production cost of about 100–120 €/MWh. In comparison, the production cost from fossil sources (gas, oil) is about 60–70 €/MWh. Simulation results show that a KiteGen capable of generating the same mean energy can be realized using 60–70 airfoils of about 500 m^2, rotating in a carousel configuration of 1500 m radius and flying up to 800 m. The resulting land usage is 8 km^2, and the energy production cost is estimated to be about 10–15 €/MWh.

All this information has been obtained from this article [10].

1.4 Aim of the present work

The objective of the project is to develop a tool that couple the aerodynamic and the structural models. This aerodynamic-structural coupling of the kites is very strong, since they are relatively flexible.

This aim is achieved with the following specific aims:

- Write a code in order to calculate the different forces acting on the kite
- Write a code for find the equilibrium position of the kite
- Combine the two codes for a program that, given some initial parameters, get find the equilibrium position of the kite as well as the forces acting on it.

In order to do so, some important assumptions are done: The rigidity of the kite is not taken into account nor the non-stationary effects.

The present work has been done as a contribution to the article [6]. The results that are going to be showed are obtained for the same kite as the authors used in this paper.

1.5 Legislation

As this is a new way of production of renewable energy, the regulations are nowadays the same regulations that apply to any other renewable energy.

It is of hoping, that in not too much time, when it can extract a real performance to this type renewable energy, a few series of procedure will be applied to regulate his activity and the space where to be able to place the kites, but at the moment, there is no special legislation that directly affects the specific activities proposed in this project.

2 Vortex Lattice Method

2.1 Introduction

Every problem in aerodynamics is started with a physical problem, and then represent the physical situation with a mathematical model. Using the solution that the mathematical model give us, something about the physical problem is deduced. Of course, skill and experience are required to carry out this sequence of steps. In particular, judgement has to be used to select the method to be used.

The process is illustrated in Figure 7, as requiring the following steps:

- Start with the real flow around the aircraft.
- Create a physical model of the flowfield, perhaps (and traditionally) considering it as an inviscid transonic flow, a boundary layer flow and a wake.
- Create the simplified mathematical model(s) to be solved.
- Carry out the numerical solution.
- Examine the results.
- Interpret the sequence of physical model, mathematical model, and numerical solution, together with the computed results to provide the final aerodynamic solution.

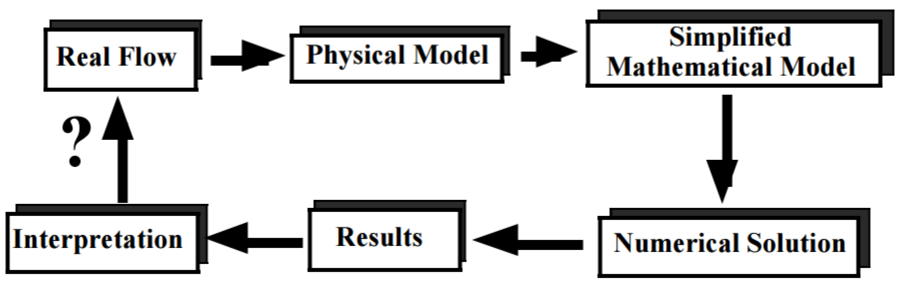


Figure 7: Typical split of functions in a CFD software system

Computational aerodynamics can provide insights in complex problems through solutions of the governing equations of fluid mechanics. The typical functions flowchart in a computational aerodynamic system is shown in Figure 8. There are three parts: geometry set-up, flow solver and post-processing. To solve the problem there are four solver options, as it can be seen in the figure. They are ordered from the most simplified flow model to the complete flow model.

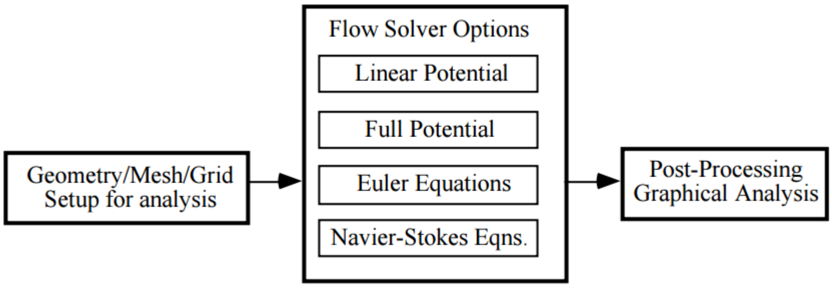


Figure 8: Steps in applying computational analysis to aerodynamics

For the inviscid, incompressible flow, the potential flow model provides reliable flowfield predictions over a wide range of conditions. The Laplace Equation is essentially an exact representation of this kind of flow.

One of the key features of Laplace's Equation is the property that allows the equation governing the flowfield to be converted from a 3D problem throughout the field to a 2D problem for finding the potential on the surface. The solution is then found using this property by distributing singularities of unknown strength over discretized portions of the surface: the panels. Therefore the flowfield solution is found by representing the surface by a number of panels, and solving a linear set of algebraic equations to determine the unknown strengths of the singularities.

Vortex Lattice Method (VLM) is a method similar to the normal Panel Method. This method is based on the idea of a vortex singularity as the solution of Laplace's equation, and it is very easy to use and capable of providing remarkable insight into wing aerodynamics and component interaction. The concept is extremely simple, but because of its purely numerical approach practical applications awaited sufficient development of computer power. Although VLM origins from the classical Prandtl lifting line theory, one key advantage of the vortex lattice method compared to lifting line theory is the ability to treat swept wings. Classical Prandtl lifting line theory is essentially correct for unswept wings, but is completely wrong for swept wings.

One interesting thing is to compare VLM with Panel method in order to see the main differences and similarities. VLM is similar to Panel method because:

1. singularities are placed on a surface.

2. the non-penetration condition is satisfied at a number of control points.

3. a system of linear algebraic equations is solved to determine singularity strengths.

VLM is different from Panel methods because:

1. it is oriented toward lifting effects, and classical formulations ignore thickness (see section 2.3.2).

2. boundary conditions are applied on a mean surface, not the actual surface (not an exact solution of Laplace's equation over a body, but embodies some additional approximations, i.e. together with the first item, we find ΔC_p, not $\Delta C_{p_{upper}}$ and $\Delta C_{p_{lower}}$).

3. singularities are not distributed over the entire surface.

4. it is oriented toward combinations of thin lifting surfaces (recall Panel Method had no limitations on thickness).

However, as it can be seen in figure 9, every method should always be compared and calibrated with experimental data to get final reliable results. This calibration is necessary because of the neglected viscous effects.

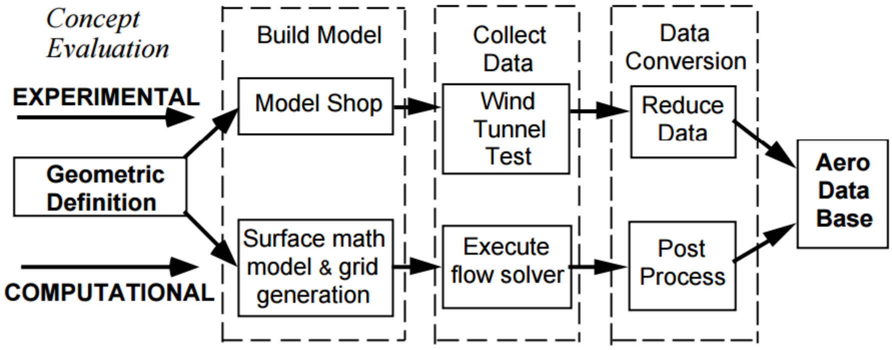


Figure 9: Analogy between computational and experimental aerodynamics

The following sections in this chapter will give more details about the physical and mathematical insights of the VLM and the implementation of this method.

2.2 Physical problem

At first look, when the air flows around a kite the generation of the aerodynamic forces may appear to be exceptionally complex, particularly the complicated three-dimensional flow field. Nevertheless, the aerodynamics forces and moments on the body are because of just two fundamental sources:

1. Pressure distribution over the body surface.

2. Shear stress distribution over the body surface.

This phenomenon enable us to separate the problem of aerodynamic forces. As the research object of this project is a kite during normal flying, this present study focuses on the pressure domain, since pressure forces are dominant in the generation of lift which the present work takes into account. This means that the aerodynamics condition will be in the incompressible domain, moreover this will be the linear potential aerodynamics domain which has small angle of attack (non-stationary effects are not considered).

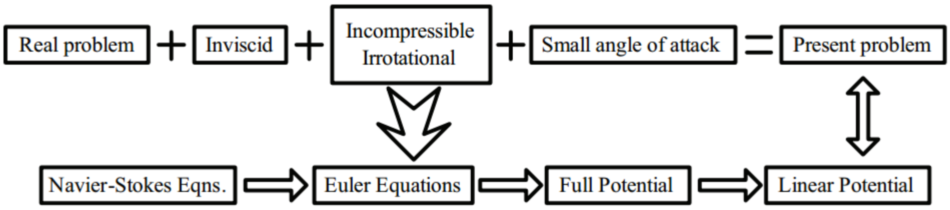


Figure 10: Simplified physical models and corresponding mathematical equations

In Figure 10 the simplified physical model for the present work is shown with the corresponding mathematical model equation. On the upper side the real problem is simplified to the present problem under four important conditions: inviscid, incompressible, irrotational and small angle of attack. On the lower side the Navier-Stokes equations are simplified to linear potential equations under these conditions.

As it well known, Linear aerodynamics only concerns the linear domain of kite behaviour. It has limitations, but it is very useful. This domain is limited to small Mach numbers and small angles of attack, hence the compressibility effects can be disregarded as well as it is ensured that the lift coefficients remain well below the stall limit.

2.3 Mathematical analysis

2.3.1 Potential flow

First starting from the irrotational flow which is defined as a flow where the vorticity is zero at every point:

$$\omega = \nabla \times \mathbf{V} = 0 \tag{2.3.1}$$

If ϕ is a scalar function, we can get the following the vector identity:

$$\nabla \times (\nabla \phi) = 0 \tag{2.3.2}$$

It means that the curl of the gradient of a scalar function is identically zero. Comparing equation 2.3.1 and 2.3.2, it can be seen that:

$$\mathbf{V} = \nabla\phi \tag{2.3.3}$$

The above equation states that for an irrotational flow, there exists a scalar function ϕ such that the velocity is given by the gradient of ϕ. We denote ϕ as the velocity potential.

The following equation is obtained from the principle of mass conservation for an incompressible flow, :

$$\nabla \cdot \mathbf{V} = 0 \tag{2.3.4}$$

With the definition of velocity potential ϕ, for a flow that is both incompressible and irrotational, equation 2.3.3 and 2.3.4 can be combined to give us:

$$\nabla \cdot (\nabla\phi) = 0 \tag{2.3.5}$$

or

$$\nabla^2\phi = 0 \tag{2.3.6}$$

Equation 2.3.6 is the famous Laplace's equation, it governs the irrotational, incompressible flow. Since the Laplace's equation is linear, Anderson [7] concludes that: *a complicated flow pattern for an irrotational, incompressible flow can be synthesized by adding together a number of elementary flows which are also irrotational and incompressible.* These different elementary flows include point/line source, point/line sink, point/line doublet and point/line vortex. The VLM method is based on the line vortices.

2.3.2 Boundary conditions

A very useful principle was obtained by Mason [15]: *cases where the linearized pressure coefficients relation is valid, thickness does not contribute to lift to first order in the velocity disturbance!*. Mason stated that after applying the linear approximation between velocity and pressure on the lower and upper surfaces of the airfoil. VLM uses the so-called "thin airfoil boundary condition". This condition includes the linearisation of the boundary condition (with the small disturbance assumption), the transfer of the boundary condition (as shown in Figure 11), and the linear approximation between velocity and pressure (with the small disturbance assumption). The importance of this analysis is that the cambered surface boundary conditions can be applied on a flat coordinate surface and result in a much more easy way to apply boundary conditions.

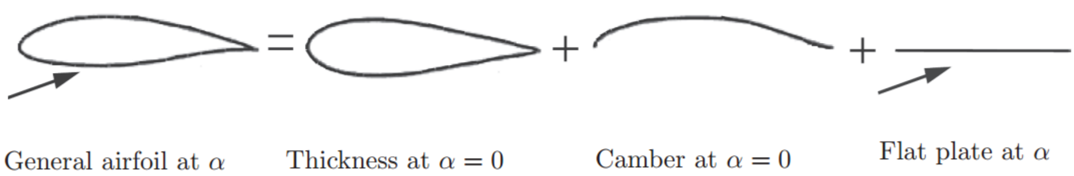


Figure 11: Decomposition of a general airfoil at certain incidence

Taking the condition that the camber effect can also be neglected (symmetrical airfoil/wing), as it going to be the case, after applying this boundary condition to the linear partial differential equation (Laplace's equation) the problem can easily be solved by including the effect of angle of attack on a flat surface.

In the following parts, in order to describe the method, the kite will be placed on the x-y plane. The boundary condition states that normal flow across the thin kite's solid surface is zero:

$$\nabla(\Phi + \Phi_\infty) = 0 \tag{2.3.7}$$

It means that the sum of the normal velocity component induced by the bound vortices of the wing w_b, by the wake w_i and by the free-stream velocity Q_∞ will be zero:

$$w_b + w_i + Q_\infty \alpha = 0 \tag{2.3.8}$$

2.3.3 Biot-Savart law

As stated before, the two-dimensional (point) vortex singularity is one of the solutions for Laplace's equation. The vortex flow is illustrated in Figure 12. The induced tangential velocity by a vortex is defined below:

$$V_\theta = \frac{\Gamma}{2\pi r} \tag{2.3.9}$$

where Γ is the field circulation strength and it is constant around the circle of radius r. Because the circulation has the same sign as the vorticity so it is positive in the clockwise direction. Here r is the radius to flow centre.

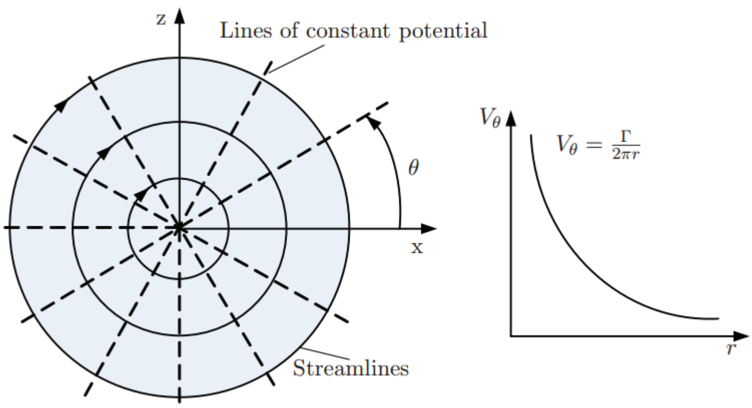


Figure 12: Point Vortex Flow

The key properties of a vortex filament are determined by the Kelvin and Helmholtz theorems. Related to these theorems an important result can be obtained: *A line (sheet)* *of vortices can support a jump in tangential velocity (i.e. a force). while the normal velocity is continuous. This means that a vortex line (sheet) can be used to represent a lifting line (surface).*

If the point vortex idea is extended to the case of a general three-dimensional vortex filament, the flowfield induced by the vortex filament can be shown in Figure 13. Biot-Savart law give us the mathematical description of the flow induced by the filament. It states that the increment of the velocity $d\mathbf{V}$ at a point p due to a segment of a vortex filament $d\mathbf{l}$ at point q is:

$$d\mathbf{V}_p = \frac{\Gamma}{4\pi} \cdot \frac{d\mathbf{l} \times \mathbf{r}_{pq}}{|\mathbf{r}_{pq}|^3} \tag{2.3.10}$$

Integrating over the entire length of the vortex filament to obtain the induced velocity:

$$dV_p = \frac{\Gamma}{4\pi} \int \frac{d\mathbf{l} \times \mathbf{r}_{pq}}{|\mathbf{r}_{pq}|^3} \tag{2.3.11}$$

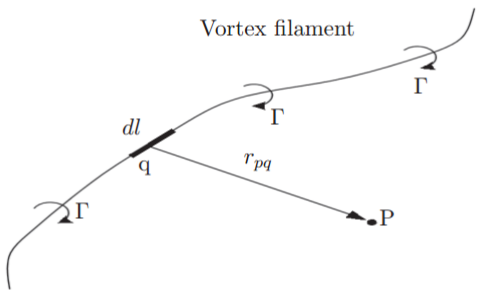


Figure 13: Vortex filament

2.3.4 Horseshoe vortex

The most important form of vortex that the vortex lattice method used is the horseshoe vortex. For the analysis the vortex is going to be placed in the x-y plane as shown in Figure 14. This vortex consist on four vortex filaments. The two trailing vortex filaments are placed parallel to the x-axis and start in infinity while the two finite vortex segments are closing the vortex ring. Normally the effects of the vortex filament AD can be neglected because of the infinite distance.The straight bound vortex segment BC models the lifting properties and the two semi-infinite trailing vortex lines model the wake. So the general expression of the induced velocity at a point by the horseshoe vortex is:

$$V = V_{BC} + V_{AB} + V_{CD} \tag{2.3.12}$$

For the finite length vortex segment BC in the horseshoe vortex, the induced velocity at certain point can be calculated using the following equation where r_1 and r_2 are the distances from this certain point to the two end points of the segment. r_0 is the length of the segment.

$$V_p = \frac{\Gamma}{4\pi} \frac{|r_1 \times r_2|}{|r_1 \times r_2|^2} \left[r_0 \cdot \left(\frac{r_1}{|r_1|} \right) - \left(\frac{r_2}{|r_2|} \right) \right] \tag{2.3.13}$$

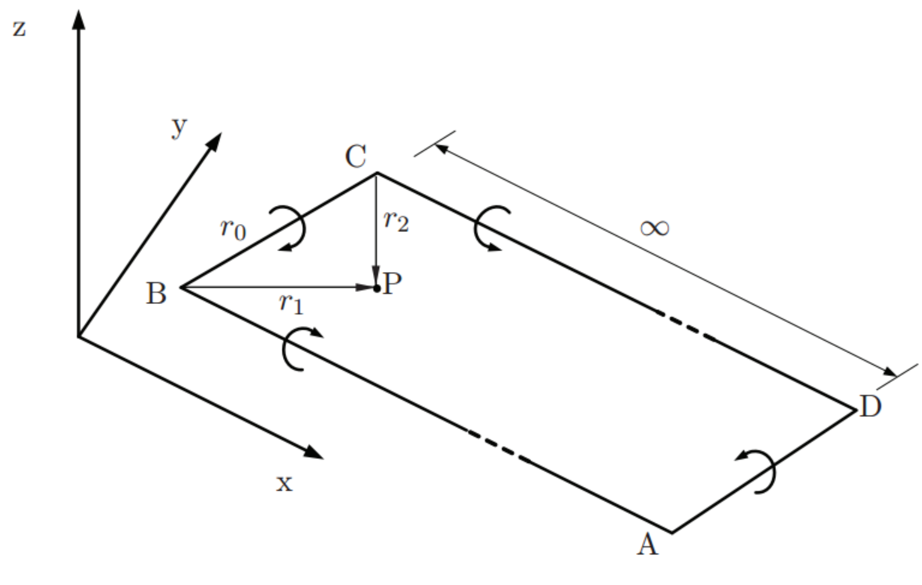


Figure 14: Horseshoe Vortex Element

2.3.5 Selection of Control Point/Vortex location

Since the horseshoe vortex is going to be used to represent a lifting surface, the following questions should be answered first: where are the vortices and the control points to be located to satisfy the surface boundary condition? By equating the lift equations from thin airfoil theory and Kutta-Joukowsky theorem, a new equation for circulation can be obtained. Then substituting the new circulation equation to the boundary condition which requires the induced velocity at the control point and the velocity at the boundary should be equal, and then the locations of the bound vortex and the control point can be determined.

The final result is called the "1/4 - 3/4 rule" which states that the vortex is located at the 1/4 chord point, and the control point is located at the 3/4 chord point. This is not a theoretical law, it is just a replacement that works well and has become a rule of thumb.

See the Appendix A for more details.

2.4 Computational Process Without Wake Vortex Interaction

2.4.1 Geometry and Flow Data

The main focus of the present work is the prediction of the lift acting on a kite as well as the final shape of the kite. So the kite geometry will be simplified to just having one part: the kite itself. Although in this study only a particular condition is considered for present the results, the code allows any flight conditions to be analysed. The following sections describes the detailed procedures on how to model the kite.

2.4.2 Discretization and Grid Generation

The kite platform is divided into elements as shown in figure 15. The bound vortex is placed at the panel quarter chord line and the control point is at the centre of the panel's three-quarter chord line according to the "1/4 - 3/4 rule". the discretization and grid generation can be proceeding as follows:

1. Divide the planform into a lattice of quadrilateral panels, and put a horseshoe vortex on each panel.

2. Place the bound vortex of the horseshoe vortex on 1/4 chord element line of each panel.

3. Place the control point on the 3/4 chord point of each panel at the midpoint in the spanwise direction.

4. Assume a flat wake.

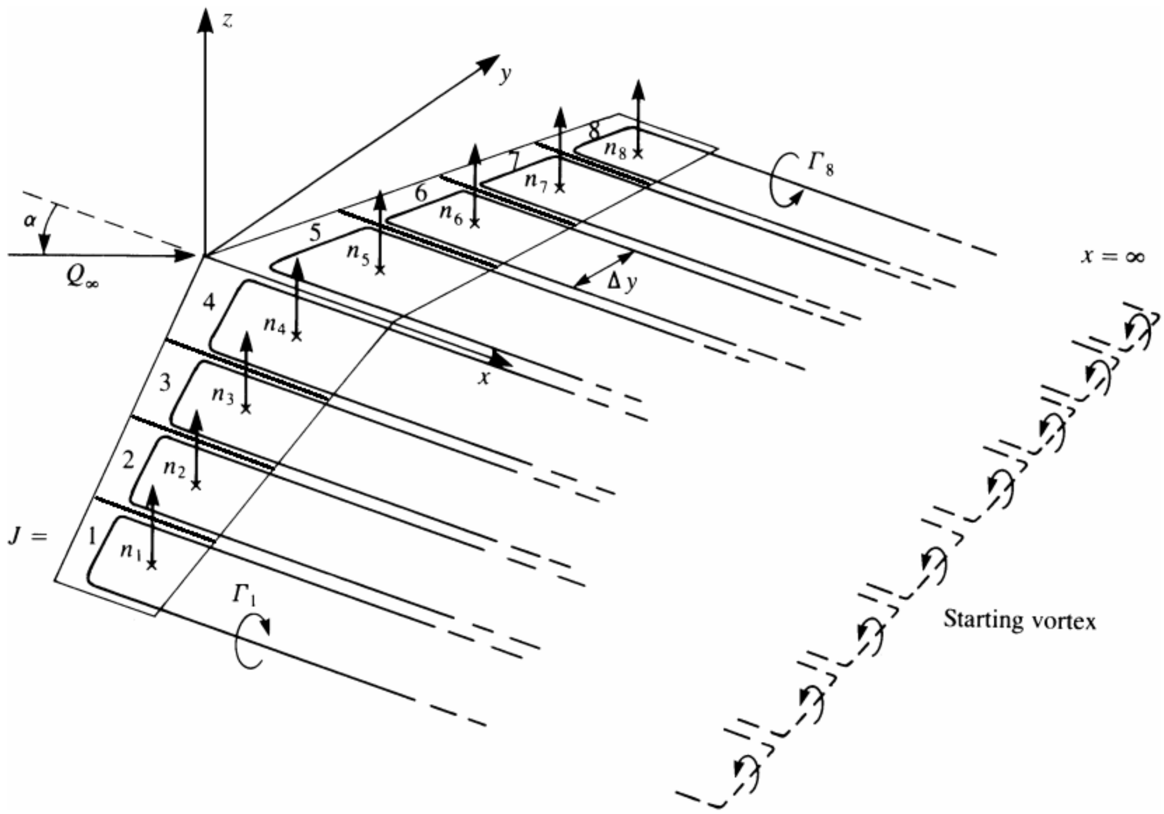


Figure 15: The horseshoe vortex lattice model for the classical vortex lattice method [13]

The strength of the vortex Γ is assumed to be constant for the horseshoe element and a positive circulation is defined as shown in the figure 15. In the figure the wing is divided into 8×1 elements, and the spanwise and chordwise counter j and i will have values between 1 $\rightarrow$ 8 and 1 respectively. The geometrical information such as the normal vector $\overrightarrow{n_j}$ of each element and the coordinates of the control points $(x_{j,i}, y_{j,i}, z_{j,i})$ are calculated at this phase. The normal $\overrightarrow{n_j}$ is calculated using three of the four points that define the element:

$$\overrightarrow{n_j} = \overrightarrow{AB} \times \overrightarrow{AC} \tag{2.4.1}$$

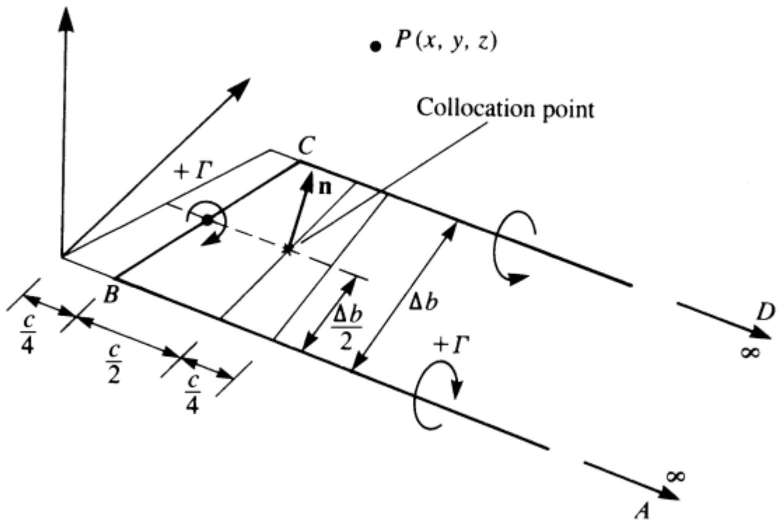


Figure 16: A spanwise horseshoe vortex element [13]

As the kite considered in the study is not contained in the x-y plane, it is required to calculate the alpha of each of the panels depending on the angle θ that the panel forms in the y-z plane. Knowing that $\overrightarrow{n_\alpha} = [\cos\alpha, 0, \sin\alpha]$:

$$\alpha_j = \frac{\pi}{2} - \arccos\left(-\frac{\overrightarrow{n_j} \cdot \overrightarrow{n_\alpha}}{\|\overrightarrow{n_j}\| \cdot \|\overrightarrow{n_\alpha}\|}\right) \tag{2.4.2}$$

2.4.3 Routine Functions

After the geometry set-up is finished, two routine function are built in the code for calculating the velocity induced by each horseshoe element. The first one is denoted

VLINE and will be used to calculate the velocity induced by the vortex lines. The second one is denoted HORSESHOE and will be used to calculate the velocity induced by the horseshoe elements.

VLINE At a certain point P, the velocity induced by vortex line AB, BC and CD as shown in figure 14 can be calculated by using equation 2.3.13. Taking the vortex line AB as an example:

$$V_p = \frac{\Gamma}{4\pi} \frac{|r_A \times r_B|}{|r_A \times r_B|^2} \left[r_0 \cdot \left(\frac{r_A}{|r_A|} \right) - \left(\frac{r_B}{|r_B|} \right) \right]$$

1. Calculate $r_A \times r_B$:

$$(r_A \times r_B)_x = (y_p - y_A)(z_p - z_B) - (z_p - z_A)(y_p - y_B)$$
$$(r_A \times r_B)_y = -(x_p - x_A)(z_p - z_B) - (z_p - z_A)(x_p - x_B)$$
$$(r_A \times r_B)_z = (x_p - x_A)(y_p - y_B) - (y_p - y_A)(x_p - x_B)$$

   The absolute value of this vector product is:

$$|r_A \times r_B|^2 = (r_A \times r_B)_x^2 + (r_A \times r_B)_y^2 + (r_A \times r_B)_z^2$$

2. Calculate the distances $|r_A|$ and $|r_B|$:

$$|r_A| = \sqrt{(x_p - x_A)^2 + (y_p - y_A)^2 + (z_p - z_A)^2}$$
$$|r_B| = \sqrt{(x_p - x_B)^2 + (y_p - y_B)^2 + (z_p - z_B)^2}$$

3. Check for singular values: In our case, this step can be skipped because the kite studied does not have swept.

4. Calculate the dot-product:

$$r_0 \cdot r_A = (x_B - x_A)(x_p - x_A) + (y_B - y_A)(y_p - y_A) + (z_B - z_A)(z_p - z_A)$$
$$r_0 \cdot r_B = (x_B - x_A)(x_p - x_B) + (y_B - y_A)(y_p - y_B) + (z_B - z_A)(z_p - z_B)$$

5. The resulting velocity components are:

$$u = K(r_A \times r_B)_x$$
$$v = K(r_A \times r_B)_y$$
$$w = K(r_A \times r_B)_z$$

where
$$K = \frac{\Gamma}{4\pi \left|r_A \times r_B\right|^2} \left[r_0 \cdot \left(\frac{r_A}{|r_A|}\right) - \left(\frac{r_B}{|r_B|}\right) \right]$$

All of the above steps are included in a subroutine that will calculate the induced velocity (u, v, w) at a point $P(x, y, z)$ as a function of the vortex line strength and its two end points coordinates:

$$(u, v, w) = VLINE(x, y, z, x_A, y_A, z_A, x_B, y_B, z_B, \Gamma) \tag{2.4.3}$$

HORSESHOE Recal equation 2.3.12:

$$V = V_{BC} + V_{AB} + V_{CD}$$

So for a point $P(x, y, z)$ in the horseshoe vortex as shown in figure 14, the induced velocity can be calculated by three applications of the vortex line routine VLINE:

$$\begin{aligned} V_{AB} &= (u_1, v_1, w_1) = VLINE(x, y, z, x_A, y_A, z_A, x_B, y_B, z_B, \Gamma) \\ V_{BC} &= (u_2, v_2, w_2) = VLINE(x, y, z, x_B, y_B, z_B, x_C, y_C, z_C, \Gamma) \\ V_{CD} &= (u_3, v_3, w_3) = VLINE(x, y, z, x_C, y_C, z_C, x_D, y_D, z_D, \Gamma) \end{aligned} \tag{2.4.4}$$

Following the small-disturbance lifting-line approach, some assumptions are used here:

$$x_A = x_D \to \infty$$
$$y_A = y_B, y_C = y_D$$

Where ∞ means that the influence of the vortex line beyond x_A or x_A is negligible, which from the practical point of view means at least twenty wing spans behind the wing is requested (In our code, a hundred wing spans are used). At this point it is possible to align the wake with the free stream by adjusting the points at $x = \infty$ which means $z_A = x_A \sin\alpha$ and $z_D = x_D \sin\alpha$, α is the angle of attack of free stream.

So the velocity induced by the three vortex segments is obtained as:

$$(u, v, w) = (u_1, v_1, w_1) + (u_2, v_2, w_2) + (u_3, v_3, w_3) \tag{2.4.5}$$

which is the HORSESHOE subroutine function:

$$(u, v, w) = HORSESHOE(x, y, z, x_A, y_A, z_A, x_B, y_B, z_B, x_C, y_C, z_C, x_D, y_D, z_D, \Gamma) \tag{2.4.6}$$

For convenience the trailing vortex wake-induced downwash $(u, v, w)^*$ will also be obtained here. This information is needed for the induced-drag computations and downwash. The influence of the trailing segment is obtained by simply omitting the influence of the bound vortex segment (u_2, v_2, w_2) from equation 2.4.5. It is automatically obtained as a by-product of the subroutine HORSESHOE.

$$(u, v, w)^* = (u_1, v_1, w_1) + (u_3, v_3, w_3) \tag{2.4.7}$$

2.4.4 Influence Coefficients

To fulfil the boundary conditions, equation 2.3.8 is specified at each of the control points as shown in figure 16. The velocity induced by the horseshoe vortex element no.1 at control point no.1 (therefore using the index '11') can be computed by using the HSHOE routine developed before:

$$(u, v, w)_{11} = HORSESHOE(x_1, y_1, z_1, x_{A1}, y_{A1}, z_{A1}, x_{B1}, y_{B1}, z_{B1}, x_{C1}, y_{C1}, z_{C1}, x_{D1}, y_{D1}, z_{D1}, \Gamma = 1)$$

Note that $\Gamma = 1$ is used to evaluate the influence coefficient due to a unit strength vortex. Similarly the velocity induced by the second vortex at the first control point will be:

$$(u, v, w)_{12} = HORSESHOE(x_1, y_1, z_1, x_{A2}, y_{A2}, z_{A2}, x_{B2}, y_{B2}, z_{B2}, x_{C2}, y_{C2}, z_{C2}, x_{D2}, y_{D2}, z_{D2}, \Gamma = 1)$$

The scanning sequence method will be used here for matrix calculation convenience in the following section. After applying the sequence method, the zero normal flow across the wing boundary condition (Equation 2.3.8) can be rewritten for the first control point as:

$$\begin{aligned}[(u, v, w)_{11}\Gamma_1 + (u, v, w)_{12}\Gamma_2 + (u, v, w)_{13}\Gamma_3 + \cdots \\ +(u, v, w)_{1K}\Gamma_K + (U_\infty, V_\infty, W_\infty)] \cdot \overrightarrow{n_1} = 0\end{aligned} \tag{2.4.8}$$

where K is equal to the number of elements. The strengths of the vortices Γ_j are not known at this phase. Establishing the same procedure for each of the collocation points results in the discretized form of the boundary condition:

$$\begin{aligned}
a_{11}\Gamma_1 + a_{12}\Gamma_2 + a_{13}\Gamma_3 + \cdots + a_{1K}\Gamma_K &= -\overrightarrow{Q_\infty} \cdot \overrightarrow{n_1} \\
a_{21}\Gamma_1 + a_{22}\Gamma_2 + a_{23}\Gamma_3 + \cdots + a_{2K}\Gamma_K &= -\overrightarrow{Q_\infty} \cdot \overrightarrow{n_2} \\
a_{31}\Gamma_1 + a_{32}\Gamma_2 + a_{33}\Gamma_3 + \cdots + a_{3K}\Gamma_K &= -\overrightarrow{Q_\infty} \cdot \overrightarrow{n_3} \\
\vdots \qquad & \qquad \vdots \\
a_{K1}\Gamma_1 + a_{K2}\Gamma_2 + a_{K3}\Gamma_3 + \cdots + a_{KK}\Gamma_K &= -\overrightarrow{Q_\infty} \cdot \overrightarrow{n_K}
\end{aligned}$$

where the influence coefficients are defined as:

$$a_{mn} \equiv (u, v, w)_{mn} \cdot \overrightarrow{n_m} \tag{2.4.9}$$

where m is in the outer loop and stands for the counter of control points, n is in the inner loop and stands for the counter of horseshoe vortex element. And both of them are from 1 to K.

And again we get a byproduct:

$$b_{mn} \equiv (u, v, w)^*_{mn} \cdot \overrightarrow{n_m} \tag{2.4.10}$$

Here b_{mn} is the normal component of the wake-induced downwash that will be used for the induced drag computation and $(u, v, w)^*$ has been given by equation 2.4.7.

2.4.5 Establish RHS Vector and Solve Linear Set of Equations

The normal velocity components of the free-stream flow $\overrightarrow{Q_\infty} \cdot \overrightarrow{n_m}$ are known and moved to the right-hand side of the equation:

$$RHS_m \equiv -(U_\infty, V_\infty, W_\infty) \cdot \overrightarrow{n_m} \tag{2.4.11}$$

The right-hand side vector is actually the normal component of the free stream, which can be computed within the outer loop of the influence coefficient computation.

Now a set of K linear algebraic equations with K unknown values Γ_m can be solved by standard matrix solution techniques:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1K} \\ a_{21} & a_{22} & \cdots & a_{2K} \\ a_{31} & a_{32} & \cdots & a_{3K} \\ \vdots & \vdots & \ddots & \vdots \\ a_{K1} & a_{K2} & \cdots & a_{KK} \end{pmatrix} \cdot \begin{pmatrix} \Gamma_1 \\ \Gamma_2 \\ \Gamma_3 \\ \vdots \\ \Gamma_K \end{pmatrix} = - \begin{pmatrix} [U_\infty \cos(\alpha_1), 0, U_\infty \sin(\alpha_1))] \cdot \overrightarrow{n_1} \\ [U_\infty \cos(\alpha_2), 0, U_\infty \sin(\alpha_2)] \cdot \overrightarrow{n_2} \\ [U_\infty \cos(\alpha_3), 0, U_\infty \sin(\alpha_3)] \cdot \overrightarrow{n_3} \\ \vdots \\ [U_\infty \cos(\alpha_K), 0, U_\infty \sin(\alpha_K)] \cdot \overrightarrow{n_K} \end{pmatrix} \tag{2.4.12}$$

The RHS vector depends on α and on θ, since the normal vectors and the alpha of each of the panels depends on both.

2.4.6 Secondary Computations

The solution of the above set of equations results in the vector $(\Gamma_1, \Gamma_2, ..., \Gamma_K)$. The lift of each bound vortex segment is obtained by using the Kutta-Joukowski theorem:

$$\Delta L_j = \rho Q_\infty \Gamma_j \Delta y_j \cos\theta_j \cos\alpha_j \tag{2.4.13}$$

where Δy_j is the panel bound vortex projection normal to the free stream (see figure 16 where the panel width $\Delta b = \Delta y$) and θ_j is the angle that forms the panel in the y-z plane (see figure 21). The induced drag computation is somewhat more complex.

$$\Delta D_j = -\rho w_{ind_j} \Gamma_j \Delta y_j \tag{2.4.14}$$

where the induced downwash w_{ind_j} at each control point is computed by summing the velocity induced by all the trailing vortex segments. This can be done by the following matrix formulation where all the b_{mn} and the Γ_j are known:

$$\begin{pmatrix} w_{ind_1} \\ w_{ind_2} \\ w_{ind_3} \\ \vdots \\ w_{ind_K} \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1K} \\ b_{21} & b_{22} & \cdots & b_{2K} \\ b_{31} & b_{32} & \cdots & b_{3K} \\ \vdots & \vdots & \ddots & \vdots \\ b_{K1} & b_{K2} & \cdots & b_{KK} \end{pmatrix} \cdot \begin{pmatrix} \Gamma_1 \\ \Gamma_2 \\ \Gamma_3 \\ \vdots \\ \Gamma_K \end{pmatrix}$$

The total lift and drag are then calculated by summing the individual horseshoe contribution:

$$Lift = \sum_{j=1}^{K} \Delta L_j \tag{2.4.15}$$

$$Induced\ Drag = \sum_{j=1}^{K} \Delta D_{ind_j} \tag{2.4.16}$$

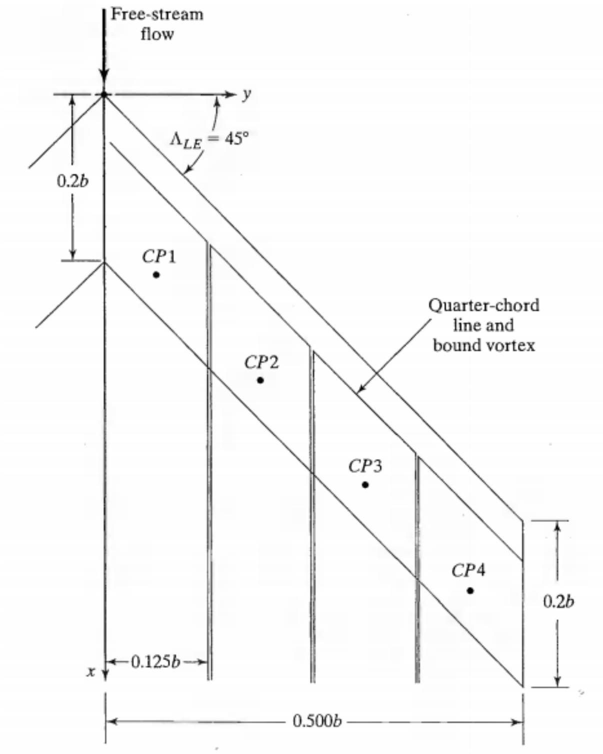
2.5 Validation of the Aerodynamic Code

This section deals with the validation of the Aerodynamic Code. Two validations are going to be performed. The first validation is executed by comparing the computational results with another VLM code called Tornado from KTH programmed by Melin [16] and also with theoretical results from Bertin & Smith [9]. And the second validation is performed by comparing the results with a 3D-wing computed by XFRL5 ®.

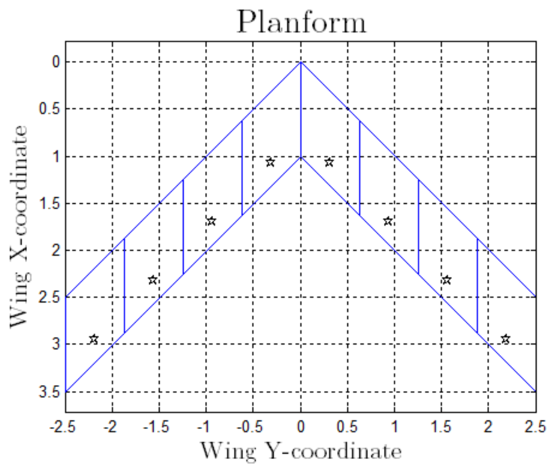
2.5.1 Tornado Code

Tornado is a Vortex Lattice Method for linear aerodynamic wing design applications in conceptual aircraft design or in aeronautical education. By modelling all lifting surfaces as thin plates, Tornado can solve for most aerodynamic derivatives for a wide range of aircraft geometries. With a very high computational speed, Tornado gives the user an immediate feedback on design changes, making quantitative knowledge available earlier in the design process. The Tornado solver solves for forces and moments, from which the aerodynamic coefficients are computed. The code is implemented in MATLAB and is being used at many universities and corporations around the world. It is still in a development phase.

The case analysed is a simple swept wing with an Aspect Ratio equal to 5 and a chord equal to 1 m.



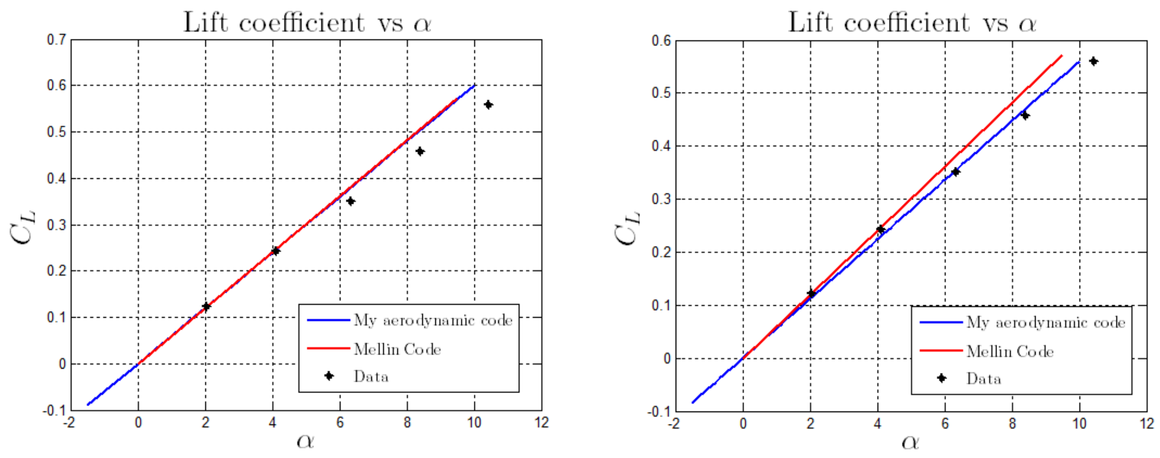
(a) Bertin & Smith Planform



(b) Matlab Planform

Figure 17: Planforms

Bertin & Smith [9] have provided a computational example of the VLM based on a swept wing presented in figure 17a. They used a total of 8 panels where the lift-curve slope will be determined and this case is also used in the validation of Tornado by Melin [16]. Figure 17b is the geometry layout of the wing in the Aerodynamic Code that is going to be validated. The calculation results of Aerodynamic code are shown in figure 18a together with the experimental results from Tornado and Bertin & Smith. It can be seen that the results from our code match perfectly the results from Tornado but not the theoretical data from Bertin & Smith. This is due to the number of panels used in the analysis. In the figure 18b it can be seen as increasing the number of panels, specifically to 891 panels, the solution matches well with the theoretical data.



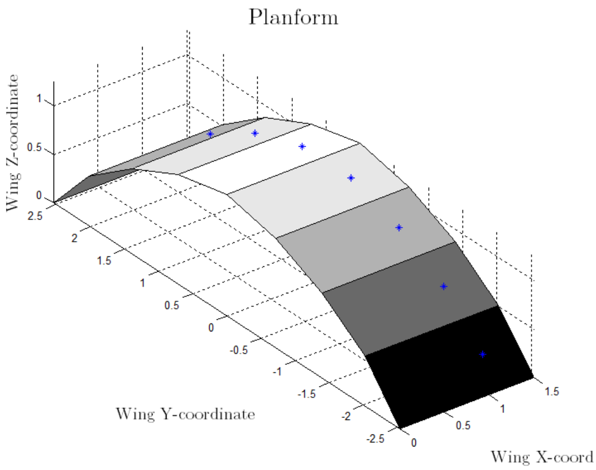
(a) (b)

Figure 18: Comparison of the theoretical and the experimental lift coefficients for the swept wing of figure 17 in a subsonic stream

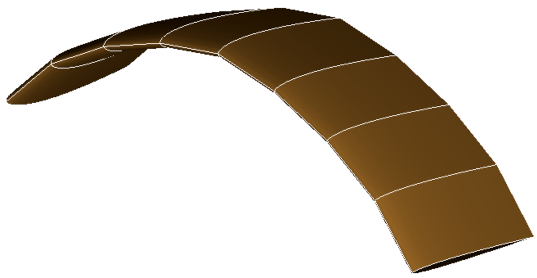
2.5.2 XFLR5

Now, the validation of the Aerodynamic Code is going to be performed using XFLR5 ®. A 3D-wing or kite is the one chosen for the comparison. This kite has an Aspect Ratio equal to 3.87, and a chord equal to 1.5 m. This is going to be the same kite used in Section 5.

As it can be seen in Figure 19 the angles of the kite in the y-z plane are 45,30, 20 and 10 degrees respectively, being a symmetric kite. The kite shown in Figure 19a is the one used in Matlab, being the kite in Figure 19b used in XFLR5. Both of them are exactly the same.



(a)



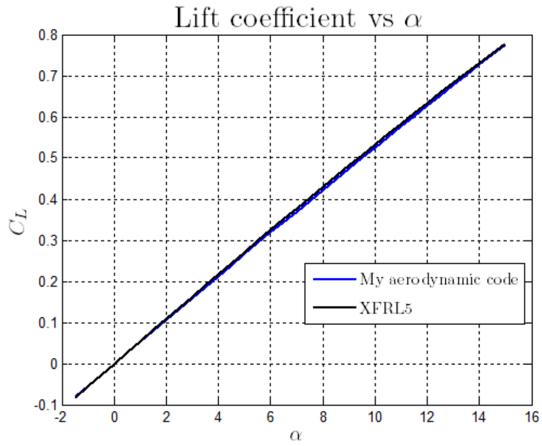
(b)

Figure 19: Planforms

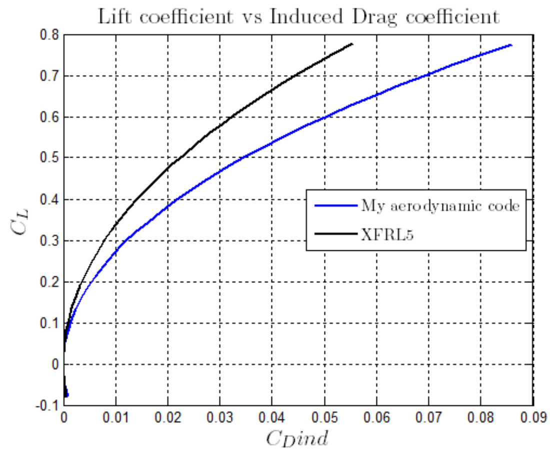
The results from the Aerodynamic Code and XFLR5 are plotted in Figure 20. As it can be seen in Figure 20a the C_{L_α} provided by the Aerodynamic Code matches well the one from XFRL5. Also the Lift coefficient versus the induced Drag coefficient is plotted in Figure 20b. It can be appreciated a different in the induced Drag. This different is due to the way that XFRL5 calculate the Induced Drag:

$$C_{D_{ind}} = \frac{C_L^2}{\pi e AR} \tag{2.5.1}$$

where e is the wing span efficiency value by which the induced drag exceeds that of an elliptical lift distribution, typically 0.85 to 0.95.



(a)



(b)

Figure 20: Comparison of the theoretical and the experimental lift and induced drag coefficients for the kite of figure 19 in a subsonic stream

3 Structural model

In this section, the structural model of the kite is going to be studied.

The kite is divided in N number of panels. Each panel has the same length. In addition there are two tethers that connect the ends of the kite with a common point, located on the ground.

There are two constraints that must be met always:

1. The geometry has to be symmetric.

2. The forces acting on each panel also have to be symmetric.

The unknown is the vector $\overrightarrow{\theta} = [\theta_0, \theta_1, \theta_2, ..., \theta_N, \theta_{N+1}]$, being in total $N+2$ unknowns. Defining $\theta > 0$ as the angle between the element panel/tether and the Y-axis(> 0)

3.1 Nomenclature

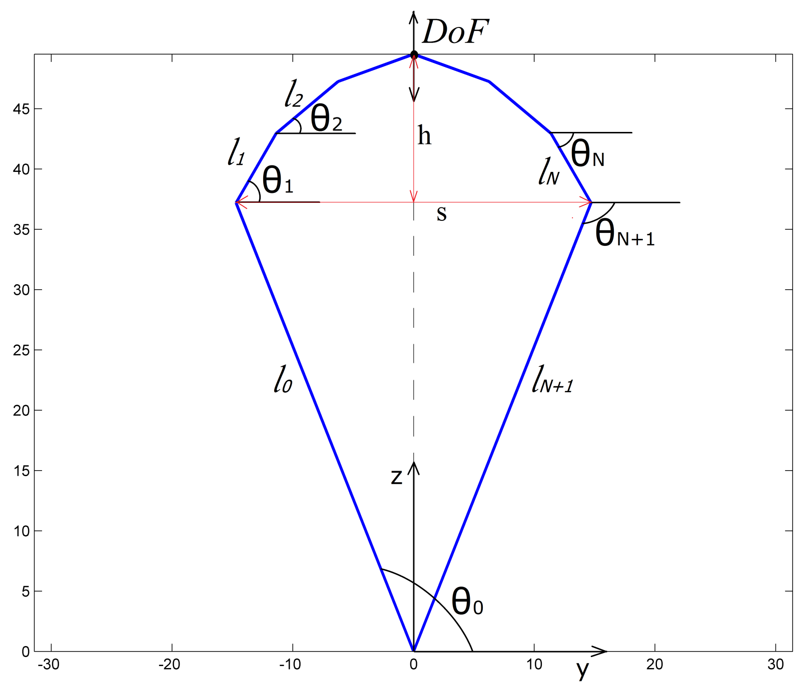


Figure 21: Nomenclature

In Figure 21 it can be seen the nomenclature used in this section. As it can be appreciated the angles are formed by the horizontal line and the panel. This produce that the angles at the y positive side of the kite are negatives. Also the distances l_0 and l_{N+1} are showed. As stated before, this two tethers are equal. Furthermore, two important parameters are defined in the figure, h and s. This parameters are used later in order to see the behaviour of the shape of the kite.

3.2 Geometric constraint / Compatibility Equations

The geometric constraint is defined on the condition imposed that the ends of the kite are joined at a common point.

$$\overrightarrow{x_0}(y_0, z_0) \equiv \overrightarrow{x_{N+2}}(y_{N+2}, z_{N+2}) \tag{3.2.1}$$

Taking into account that:

$$\overrightarrow{x_{i+1}} = \overrightarrow{x_i} + \Delta\overrightarrow{x_i} \qquad\qquad i = [0, N+1] \tag{3.2.2}$$

where $y_{i+1} = y_i + \Delta y_i$ and $z_{i+1} = z_i + \Delta z_i$

Using the Equation 3.2.1 it is possible to demonstrate that:

$$\overrightarrow{x_{N+2}} = \overrightarrow{x_0} + \sum_{0}^{N+1} \Delta\overrightarrow{x_i} \qquad \rightarrow \qquad \Delta\overrightarrow{x_i} = 0 \tag{3.2.3}$$

Therefore the equations are:

$$\begin{aligned} \sum_{0}^{N+1} \Delta y_i &= \sum_{0}^{N+1} l_i \cos(\theta_i) = 0 \\ \sum_{0}^{N+1} \Delta z_i &= \sum_{0}^{N+1} l_i \sin(\theta_i) = 0 \end{aligned} \tag{3.2.4}$$

From this two equations, the angles θ_0 and θ_{N+1} can be obtained. It is well known that this angles are equal in absolute value.

3.3 Steady state

It is assumed that the aerodynamic forces (F_j) act in the centre of each panel (P_j).

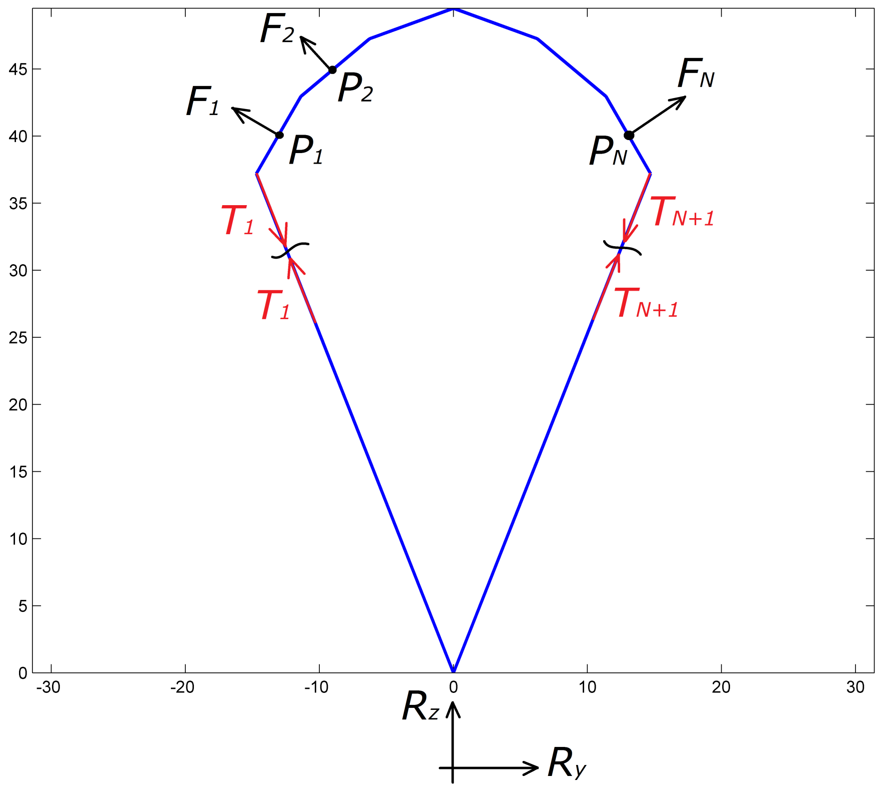


Figure 22: Forces and Tensions

Definition of the normal vectors and the external forces:

$$\overrightarrow{n_j} = n_j^y \overrightarrow{e_y} + n_j^z \overrightarrow{e_z} \tag{3.3.1}$$

where $n_j^y = -\sin(\theta_j)$ and $n_j^z = \cos(\theta_j)$. So:

$$\overline{n_j} = -\sin(\theta_j)\overrightarrow{e_y} + \cos(\theta_j)\overrightarrow{e_z} \tag{3.3.2}$$

Then:

$$\overrightarrow{F_j} = \left|\overrightarrow{F}_j\right| (-\sin(\theta_j)\overrightarrow{e_y} + \cos(\theta_j)\overrightarrow{e_z}) \tag{3.3.3}$$

3.3.1 Calculation of Reactions and Tensions $T_1(\overline{\theta})$ and $T_{N+1}(\overline{\theta})$:

$$\overrightarrow{R} = R_y\overrightarrow{e_y} + R_z\overrightarrow{e_z} \tag{3.3.4}$$

$$\overrightarrow{F_{ext}} = \sum_{1}^{N} \overrightarrow{F_j} = (-\sum_{1}^{N} \left|\overrightarrow{F}_j\right| \sin(\theta_j))\overrightarrow{e_y} + (\sum_{1}^{N} \left|\overrightarrow{F}_j\right| \cos(\theta_j))\overrightarrow{e_z} \tag{3.3.5}$$

Using the well-known equations of the Steady state:

$$\sum \overrightarrow{F}_{ext} = \overrightarrow{0} \tag{3.3.6}$$

$$\begin{aligned} \overrightarrow{j}: &\qquad R_y - \sum_{j=1}^{N} \left|\overrightarrow{F}_j\right| \sin(\theta_j) = 0 \\ \overrightarrow{k}: &\qquad R_z + \sum_{j=1}^{N} \left|\overrightarrow{F}_j\right| \cos(\theta_j) = 0 \end{aligned} \tag{3.3.7}$$

Therefore, the reactions will come given in the following way:

$$\begin{aligned} R_y &= \sum_{j=1}^{N} \left|\overrightarrow{F}_j\right| \sin(\theta_j) = 0 \\ R_z &= -\sum_{j=1}^{N} \left|\overrightarrow{F}_j\right| \cos(\theta_j) = 0 \end{aligned} \tag{3.3.8}$$

Known the reactions, it will proceed to the calculation of the tensions $T_1(\overrightarrow{\theta})$ and $T_{N+1}(\overrightarrow{\theta})$:

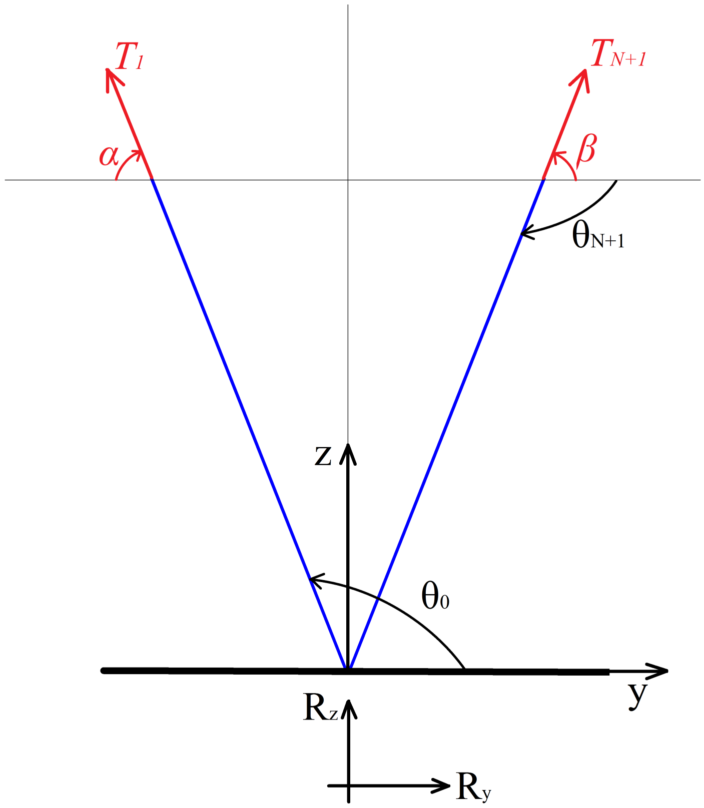


Figure 23: Tensions

$$\overrightarrow{R} = R_y \overrightarrow{e_y} + R_z \overrightarrow{e_z} \tag{3.3.9}$$

$$\begin{aligned} \overrightarrow{T_1} &= T_1^y \overrightarrow{e_y} + = T_1^z \overrightarrow{e_z} \\ \overrightarrow{T_{N+1}} &= T_{N+1}^y \overrightarrow{e_y} + = T_{N+1}^z \overrightarrow{e_z} \end{aligned} \tag{3.3.10}$$

where $\alpha > 0$ and $\beta > 0$.

$$\begin{aligned} {T_1}^y &= -\left|\overrightarrow{T}_1\right| \cos(\alpha) \\ {T_1}^z &= \left|\overrightarrow{T}_1\right| \sin(\alpha) \end{aligned} \tag{3.3.11}$$

$$\begin{aligned} {T_{N+1}}^y &= \left|\overrightarrow{T}_{N+1}\right| \cos(\beta) \\ {T_{N+1}}^z &= \left|\overrightarrow{T}_{N+1}\right| \sin(\beta) \end{aligned} \tag{3.3.12}$$

Taking into account that:

$$\begin{aligned} \alpha + \theta_0 = \pi \qquad &\rightarrow \qquad \alpha = \pi - \theta_0 \\ \beta + |\theta_{N+1}| = \pi \qquad &\rightarrow \qquad \beta = \pi - |\theta_{N+1}| \end{aligned} \tag{3.3.13}$$

Applying the following trigonometric equations:

$$\begin{aligned} \sin(x \pm y) &= \sin(x) \cdot \cos(y) \pm cos(x) \cdot \sin(y) \\ \cos(x \pm y) &= \cos(x) \cdot \cos(y) \mp sin(x) \cdot \sin(y) \end{aligned} \tag{3.3.14}$$

It is obtained that:

$$\begin{aligned} \overrightarrow{T_1} &= |T_1| \cdot [\cos(\theta_0)\overrightarrow{e_y} + \sin(\theta_0)\overrightarrow{e_z}] \\ \overrightarrow{T_{N+1}} &= |T_{N+1}| \cdot [-\cos(|\theta_{N+1}|)\overrightarrow{e_y} + \sin(|\theta_{N+1}|)\overrightarrow{e_z}] \end{aligned} \tag{3.3.15}$$

Applying again the Equation 3.3.6:

$$\begin{aligned} \overrightarrow{j}: \qquad & R_y + \left|\overrightarrow{T}_1\right| \cos(\theta_0) - \left|\overrightarrow{T}_{N+1}\right| \cos(\theta_{N+1}) = 0 \\ \overrightarrow{k}: \qquad & R_z + \left|\overrightarrow{T}_1\right| \sin(\theta_0) + \left|\overrightarrow{T}_{N+1}\right| \sin(\theta_{N+1}) = 0 \end{aligned} \tag{3.3.16}$$

Therefore it is found that:

$$\begin{Bmatrix} T_1 \\ T_{N+1} \end{Bmatrix} = \begin{bmatrix} \cos(\theta_0) & -\cos(\theta_{N+1}) \\ \sin(\theta_0) & \sin(\theta_{N+1}) \end{bmatrix}^{-1} \cdot \begin{Bmatrix} -\sum_{j=1}^{N} F_j \sin(\theta_j) \\ \sum_{j=1}^{N} F_j \cos(\theta_j) \end{Bmatrix} \tag{3.3.17}$$

Given the conditions of symmetry there happens that $|\theta_0| = |\theta_{N+1}|$. Solving the system of equations presented in 3.3.17, the following solution is obtained:

$$T_1(\overrightarrow{\theta}) = \frac{1}{\sin(\theta_0 + |\theta_{N+1}|)} \left[\cos(|\theta_{N+1}|) \sum_{j=1}^{N} F_j \cos(\theta_j) - \sin(|\theta_{N+1}|) \sum_{j=1}^{N} F_j \sin(\theta_j) \right]$$
$$T_{N+1}(\overrightarrow{\theta}) = \frac{1}{\sin(\theta_0 + |\theta_{N+1}|)} \left[\cos(\theta_0) \sum_{j=1}^{N} F_j \cos(\theta_j) + \sin(\theta_0) \sum_{j=1}^{N} F_j \sin(\theta_j) \right] \tag{3.3.18}$$

3.3.2 Calculation of Tensions in each panel

Known the tensions $T_1(\overrightarrow{\theta})$ and $T_{N+1}(\overrightarrow{\theta})$ already the N remaining equations can be obtained applying the condition of balance in every panel:

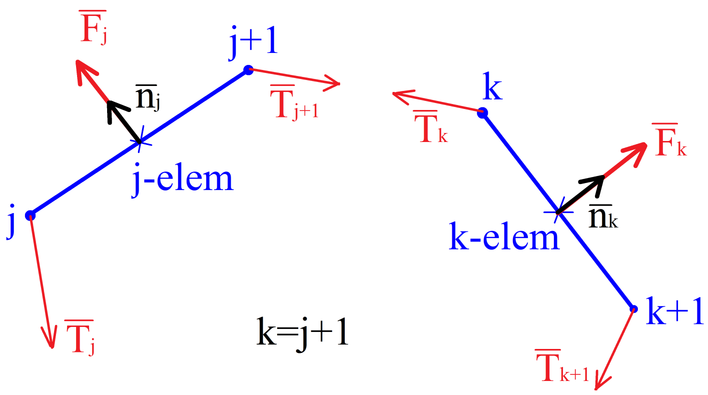


Figure 24: Forces acting on each panel

Being j and k two consecutive panels, It can be observed that $\overrightarrow{T_k} = -\overrightarrow{T_{j+1}}$.

In a k-element it is known that:

$$\overrightarrow{F_k} = F_k^y \overrightarrow{e_y} + F_k^z \overrightarrow{e_z}$$
$$\overrightarrow{T_k} = -[T_{j+1}^y \overrightarrow{e_y} + T_{j+1}^z \overrightarrow{e_z}] \tag{3.3.19}$$

Applying the equilibrium of forces, it is obtained that:

$$\overrightarrow{T_{k+1}} = -(\overrightarrow{F_k} + \overrightarrow{T_k}) \tag{3.3.20}$$

As the value of T_1 and θ_0 are known, therefore all the internal forces are going to be obtained as a function of the unknown vector $\overrightarrow{\theta}$.

3.3.3 Calculation of Moments in each panel

As it is known, there are $N+2$ unknowns. Two of them come from the geometrical constraint imposed before (Equation 3.2.4). The rest of the equations are obtained applying momentum equilibrium in each panel.

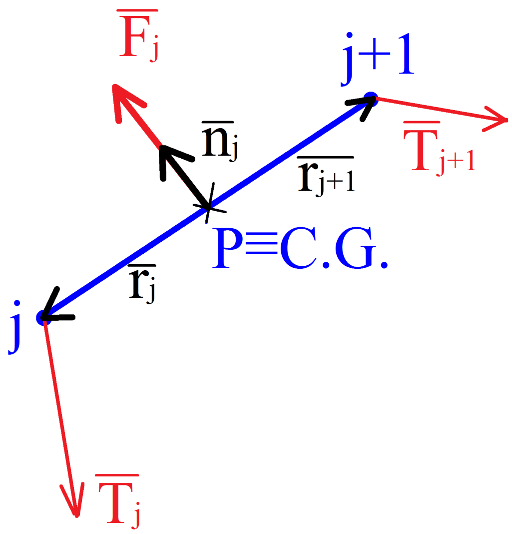


Figure 25: Moments in each panel

In the Figure 25 it can be observed that:

$$\begin{aligned} \overrightarrow{x_p} &= \frac{1}{2}(\overrightarrow{x_{j+1}} + \overrightarrow{x_j}) \qquad & \overrightarrow{r_j} &= \overrightarrow{x_j} - \overrightarrow{x_p} \\ \overrightarrow{x_{j+1}} &= \overrightarrow{x_j} + \Delta\overrightarrow{x_j} \qquad & \overrightarrow{r_{j+1}} &= \overrightarrow{x_{j+1}} - \overrightarrow{x_p} \end{aligned} \tag{3.3.21}$$

So:

$$\overrightarrow{r_j} = -\overrightarrow{r_{j+1}} = -(\frac{\Delta\overrightarrow{x_j}}{2}) \tag{3.3.22}$$

After all of these calculation, the momentum equilibrium can be applied:

$$\sum \overrightarrow{M_j}^p = \overrightarrow{0} \quad \rightarrow \quad \overrightarrow{k}: \quad [\overrightarrow{r_j} \times \overrightarrow{T_j}] + [\overrightarrow{r_{j+1}} \times \overrightarrow{T_{j+1}}] = \overrightarrow{0} \tag{3.3.23}$$

Therefore, knowing that:

$$\begin{aligned}
\overrightarrow{r_j} &= -\overrightarrow{r_{j+1}} = -(\frac{\Delta\overrightarrow{x_j}}{2}) \\
\Delta\overrightarrow{x_j} &= [l_j \cos(\theta_j), l_j \sin(\theta_j)] \\
\overrightarrow{T_{j+1}} &= T^y_{j+1}\overrightarrow{e_y} + T^z_{j+1}\overrightarrow{e_z} \\
\overrightarrow{T_j} &= T^y_j\overrightarrow{e_y} + T^z_j\overrightarrow{e_z}
\end{aligned} \tag{3.3.24}$$

It should be kept in mind that the sign of T_j is changing at every panel.

The N remaining equations 5 can be obtained applying momentum equilibrium:

$$\frac{l_j}{2}\cos(\theta_j)[T^z_{j+1} - T^z_j] - \frac{l_j}{2}\sin(\theta_j)[T^y_{j+1} - T^y_j] = 0 \qquad j = \{1, N\} \tag{3.3.25}$$

4 Computational method

As it was shown in last section, it is going to be needed a method in order to solve a system of $N + 2$ non-linear equations, being N the number of panels in which de kite is divided. In order to solve that system the Newton-Raphson method is chosen. Furthermore, a little modification to the method is done for avoid overshoots and reach the solution little by little.

Firstly, the input parameters that the user needs to enter in the program will be showed. Then the method used to solve the system of non-linear equations is described. Moreover, the errors used and the flowchart followed by the process are also described.

4.1 Input parameters

The input parameters that needed are showed in the table 3. Two very important parameters are the number of panels and γ. Increase the number of panels means that the system of equations is much more difficult to solve, as is increasing the number of non-linear equations. As it gonna be described in section 4.2, Newton-Raphson's method can not ensure convergence, so this parameter will be of vital importance.

In addition, the γ parameter may very crucial in reaching or not the convergence of the system. This parameter prevents overshoots occur and ensures that the solution is achieved little by little. At some moment of the iteration the solution may become unstable, and it will be this parameter to prevent this happening.

4.2 Newton-Raphson Method [3]

It can be done an extreme, but wholly defensible, statement: There are no good, general methods for solving systems of more than one non-linear equation. Furthermore, it is not hard to see why (very likely) there never will be any good. Consider the case of two dimensions, where we want to solve simultaneously:

Inlet conditions	Iteration loop Parameters
Wind velocity (U_∞)	Maximum number of iterations
Mach number (M_∞)	Tolerance of the errors
Angle of attack (α)	Angle variation for calculate the Jacobian ($\Delta\theta$)
Kite height with respect to sea level	Parameter used to control the iterative process (γ)
Kite parameters	
Number of panels	
Aspect Ratio	
Chord	
Length of the tethers (l_0 or l_{N+1})	

Table 3: Input Parameters

$$\begin{aligned} f(x,y) &= 0 \\ g(x,y) &= 0 \end{aligned} \tag{4.2.1}$$

The functions f and g are two arbitrary functions, each of which has zero contour lines that divide the (x, y) plane into regions where their respective function is positive or negative. These zero contour boundaries are of interest to us. The solutions that we seek are those points (if any) that are common to the zero contours of f and g (see Figure 26). Unfortunately, the functions f and g have, in general, no relation to each other at all. There is nothing special about a common point from either $f's$ point of view, or from $g's$. In order to find all common points, which are the solutions of our non-linear equations, we will (in general) have to do neither more nor less than map out the full zero contours of both functions.

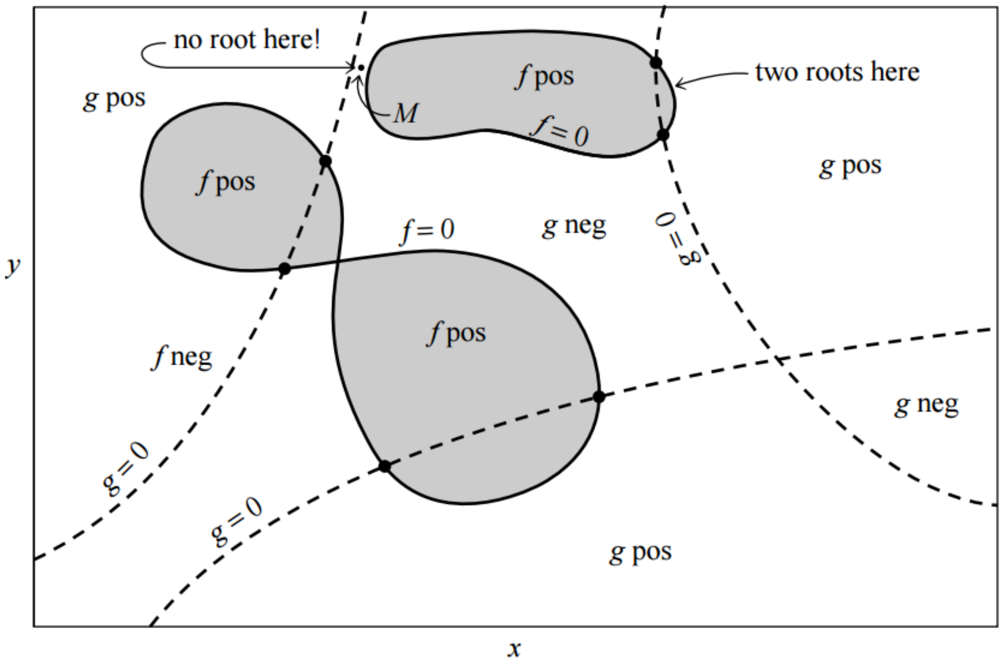


Figure 26: Solution of two nonlinear equations in two unknowns. Solid curves refer to $f(x, y)$, dashed curves to $g(x, y)$. Each equation divides the (x, y) plane into positive and negative regions, bounded by zero curves. The desired solutions are the intersections of these unrelated zero curves. The number of solutions is a priori unknown [17]

For problems in more than two dimensions, it is needed to find points mutually common to N unrelated zero-contour hyper surfaces, each of dimension $N - 1$. That root finding becomes virtually impossible without insight. Generally additional information has to be used, specific to the particular problem, to answer such basic questions as, "Do I expect a unique solution?" and "Approximately where?".

In this section will be discussed the simplest multidimensional root finding method, Newton-Raphson. This method gives you a very efficient means of converging to a root, if you have a sufficiently good initial guess. It can also spectacularly fail to converge, indicating (though not proving) that your putative root does not exist nearby. Later it is going to be discussed more sophisticated implementations of the Newton-Raphson method, which try to improve on Newton-Raphson's poor global convergence.

A typical problem gives N functional relations to be zeroed, involving variables $x_i, i =$

$1, 2, ..., N$:

$$F_i(x_1, x_2, ..., x_N) = 0 \qquad i = 1, 2, ..., N \tag{4.2.2}$$

We let $\overrightarrow{x}$ denote the entire vector of values x_i and $\overrightarrow{F}$ denote the entire vector of functions F_i. In the neighbourhood of $\overrightarrow{x}$, each of the functions F_i can be expanded in Taylor series:

$$F_i(\overrightarrow{x} + \delta\overrightarrow{x}) = F_i(\overrightarrow{x}) + \sum_{j=1}^{N} \frac{\delta F_i}{\delta x_j} \delta x_j + O(\delta\overrightarrow{x}^2) \tag{4.2.3}$$

The matrix of partial derivatives appearing in equation (9.6.3) is the *Jacobian* matrix J:

$$J_{ij} \equiv \frac{\delta F_i}{\delta x_j} \tag{4.2.4}$$

In matrix notation equation 4.2.3 is

$$\overrightarrow{F}(\overrightarrow{x} + \delta\overrightarrow{x}) = \overrightarrow{F}(\overrightarrow{x}) + J \cdot \delta\overrightarrow{x} + O(\delta\overrightarrow{x}^2) \tag{4.2.5}$$

By neglecting terms of order $\delta\overrightarrow{x}^2$ and higher and by setting $\overrightarrow{F}(\overrightarrow{x} + \delta\overrightarrow{x}) = 0$, we obtain a set of linear equations for the corrections $\delta\overrightarrow{x}$ that move each function closer to zero simultaneously, namely

$$J \cdot \delta\overrightarrow{x} = -\overrightarrow{F} \tag{4.2.6}$$

The corrections are then added to the solution vector,

$$\overrightarrow{x_{new}} = \overrightarrow{x_{old}} + \delta\overrightarrow{x} \tag{4.2.7}$$

and the process is iterated to convergence.

4.2.1 Relaxation Parameter

A parameter γ is introduced in the Newton-Raphson method in order to avoid overshoots.

$$\theta_{next} = \gamma\theta_{new} + (1-\gamma)\theta_{old} \tag{4.2.8}$$

This parameter γ is going to have a value around 10^{-4}. This value is determined by trial and error, and can be modified for other cases, e.g. the number of panels in which the kite is divided is different.

4.3 Errors

Two kinds of error are used in order to see how the solution is varying and when the error becomes an acceptable value. This errors are L^2 norm and L_∞ norm. Mathematically a norm is a total size or length of all vectors in a vector space or matrices. For simplicity, we can say that the higher the norm is, the bigger the (value in) matrix or vector is.

The most popular of all norm is the L^2 norm. It is used in almost every field of engineering and science as a whole. Following the basic definition, L^2 norm is defined as:

$$\xi_{L2} = \left(\sqrt{\frac{1}{N+2}\sum_{i=1}^{N+1}\frac{\left|\overrightarrow{\theta}^{i+1} - \overrightarrow{\theta}^{i}\right|}{\left|\overrightarrow{\theta}^{i}\right|}}\right)^2 \tag{4.3.1}$$

The L^2 norm is well known as a Euclidean norm, which is used as a standard quantity for measuring a vector difference.

The infinity norm (L_∞ norm) is defined as the maximum of the absolute values of its components:

$$\xi_{inf} = \max\left(\frac{\left|\overrightarrow{\theta}^{i+1} - \overrightarrow{\theta}^{i}\right|}{\left|\overrightarrow{\theta}^{i}\right|}\right) \tag{4.3.2}$$

4.4 Flowchart

In this section the flowchart that the program will follow is presented. Moreover, an study on how the Jacobian is calculated will be performed.

4.4.1 Jacobian

The so called Function-1 is responsible for calculating the equations for a particular $\overrightarrow{\theta}$, as well as the lift on each of the panels. In order to do so, all the methods that have been explained in previous sections are used. The result obtained from Function-1 is $\overrightarrow{F}$. This $\overrightarrow{F}$ is a column vector composes by $N+2$ elements.

The Function-2 is responsible for recalculates the $N+2$ equations using the lift calculated above, but in this case, using another $\overrightarrow{\theta}$. This new $\overrightarrow{\theta}$ is obtained varying one of its elements, one per iteration, summing a particular $\Delta\theta$ that is chosen by trial and error. This $\Delta\theta$ is of the order of 10^{-3}.

In each iteration, one column of the Jacobian is calculated using the equation 4.4.1.

$$J_{ij} \equiv \frac{\delta F_i}{\delta \theta_j} = \frac{\overrightarrow{F_{\Delta\theta}} - \overrightarrow{F}}{\Delta\theta_j} \tag{4.4.1}$$

After the Jacobian is calculated, a new $\overrightarrow{\theta}$ is obtained using the equations 4.2.7 and 4.2.8. Then, two variables are checked, the number of iterations and the errors. First the number of iterations is compared with the maximum number of iterations. If this number is smaller than the one compared, the process continue, if not the process is stopped and a message to the user is shown saying that a solution cannot be reached. After that, the second comparison arises. In this check the errors are compared with the tolerances. If they are smaller the process stop and the solution is reached, if not, the value of θ is actualized and the process continue until convergence.

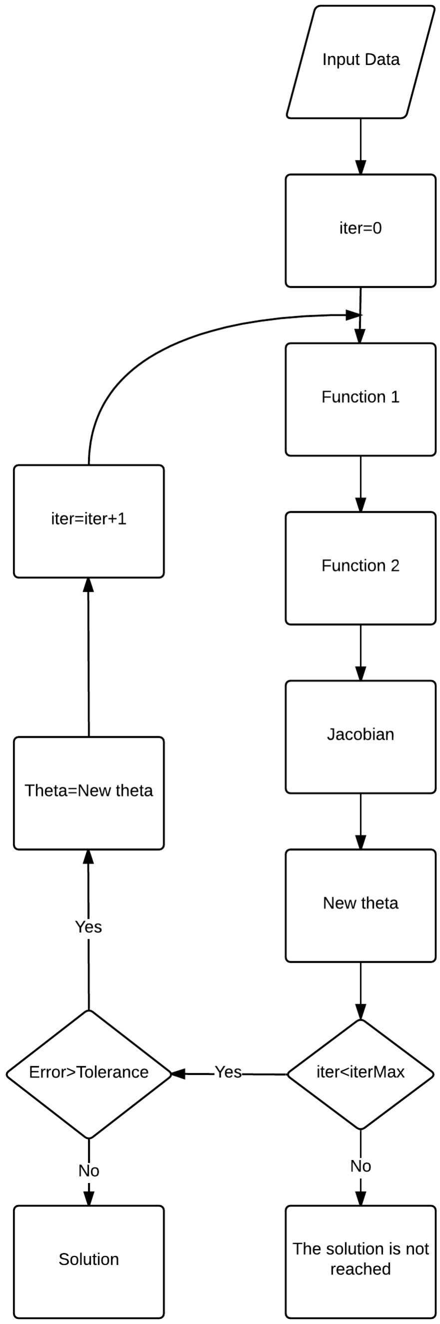


Figure 27: Flowchart

5 Results

In this section all the results of the program are going to be shown and discussed. It is important to keep in mind that the stiffness of the kite is not taken into account. This is a very huge assumption but the results will give us a first sight of the forces acting on the kite and the final shape of it.

The input parameter for this practical case are the following, the same as in the article [6]:

Kite Parameters		**Loop Parameters**	
Nº Panels	4	Max. Iterations	100000
Aspect Ratio	3.87	$Tolerance_{L_2}$	0.1
Chord	1.5 m	$Tolerance_{inf}$	0.1
Tether Length	100 m	$\Delta\theta$	0.0005
		γ	0.00005
Inlet Conditions			
Kite height with respect to sea level	750 m		
U_{inf}	14		
M_{inf}	0		

Table 4: Input Parameters for the practical case

It is also remarkable that the non-stationary effects are not taken into account. Keeping all this assumptions in mind, it is going to be shown some graphs in which the variation of Lift coefficient and the induced Drag coefficient are plotted as well as other parameters to see the variation of the kite with respect to α.

5.1 Kite geometry and forces

In the first plot, the geometry of the kite as well as the forces acting on each of the panels are plotted for a particular case of $\alpha = 5°$. It can be appreciated that the forces on the two lateral panels, that are almost perpendicular to the y-axis, are practically zero ($0.03\ N$). However, the other two forces have a value of $82N$. As it can be appreciated both the kites and the forces acting on it are symmetrical.

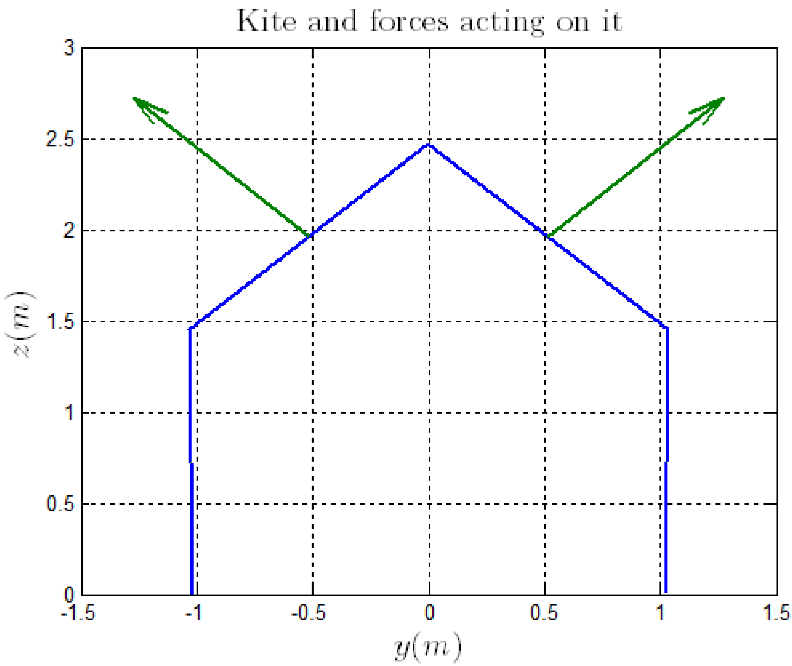
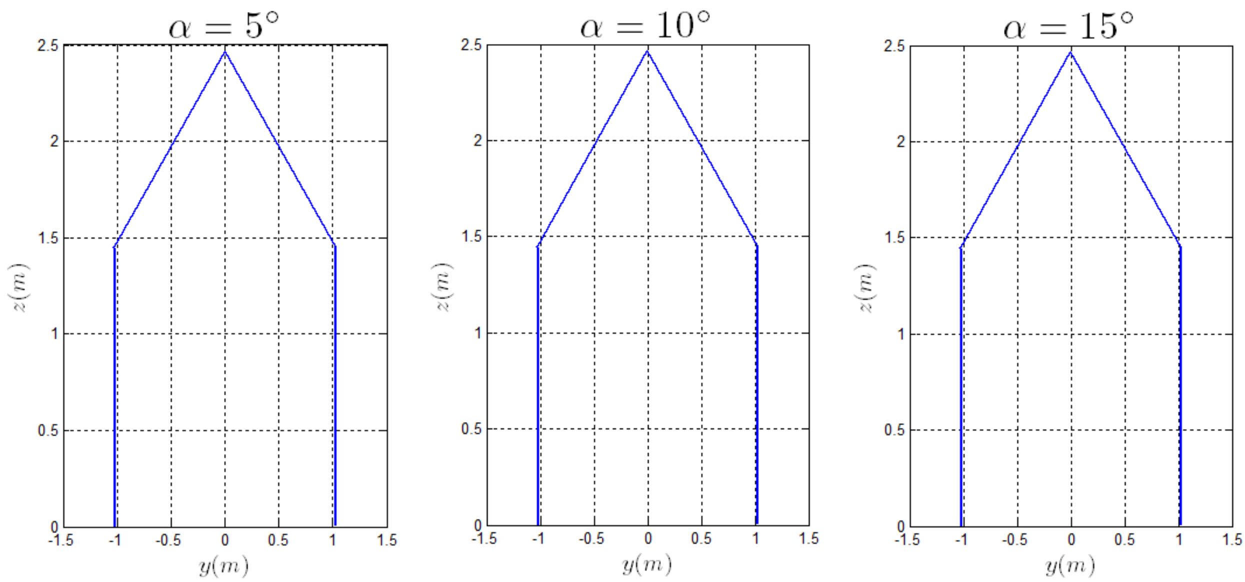


Figure 28: Geometry of the kite

It is also important to have an idea on how the geometry of the kite is changing by change the angle of attack. The figure 29 shows the kite geometry for 5, 10 and 15 degrees respectively. It can be appreciated that the kite geometry remains nearly constant while the forces acting on each panel are not equal in the three cases. This is due to the reason that the rigidity of the kite is not taken into account. As it will be explained in section 6, applying rigidity to the kite means that the forces acting on the kite will modify the shape of it.



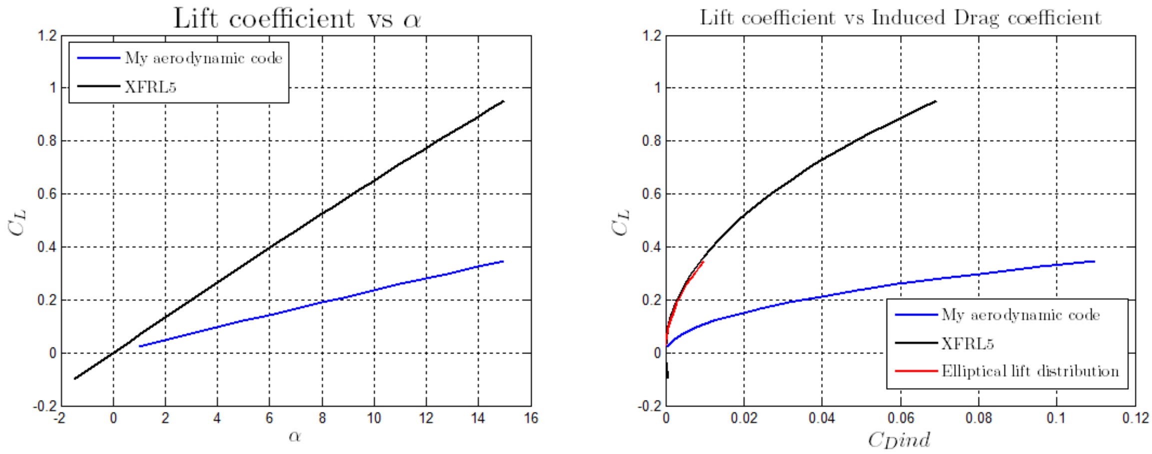
(a) (b) (c)

Figure 29: Geometry of the kite for three angles of attack

5.2 Aerodynamic coefficients

After this plots, some graphs about the aerodynamic coefficients are going to be showed. In them some comparison are done in order to see how good our kite is. In the two plots our results are compared with a flat wing of the same Aspect Ratio obtained with XFLR5. Furthermore, in figure 30b the polar is also plotted assuming an elliptical lift distribution.

As it can be seen in figure 30a the $C_{L\alpha}$ of the two kites are pretty different. Exactly, the one calculated with XFLR5 is equal to $C_{L\alpha} = 0.0333$, while our kite has a $C_{L\alpha} = 0.0240$.



(a) (b)

Figure 30: Aerodynamic coefficients

5.3 Kite parameters

Now, some plots of the parameters define at Figure 21 are showed in order to see how the kite behaves for a range of angles of attack. This parameters are h, that is the minimum vertical distance from the upper point of the kite to the line that connects the first and the last points of the kite, s which is the horizontal distance between the first and the last point of the kite, and θ_0, that have been already defined.

It can be see in the figure that the three parameters remains almost constant. This means that the shape of the kite also remains almost the same for every angle of attack. As explained before, this is due to the reason that the rigidity of the kite is not taken into account.

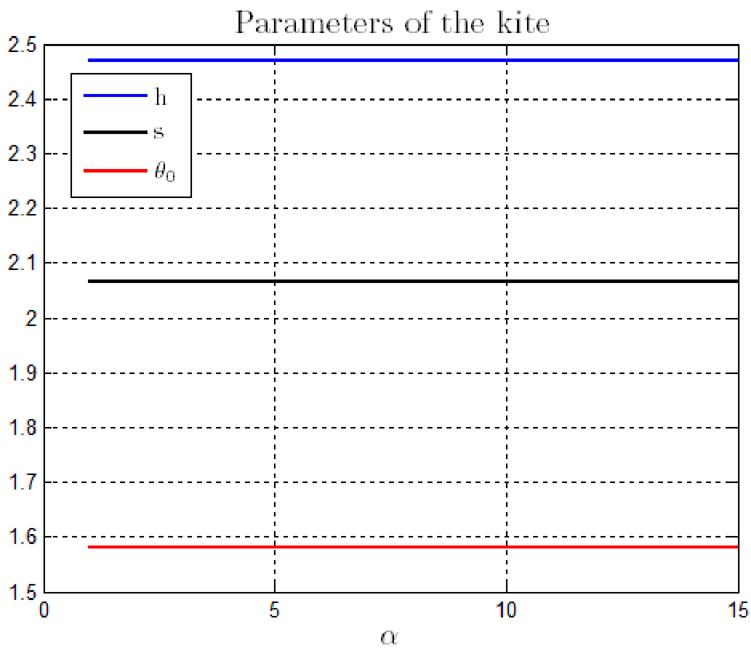


Figure 31: Kite parameters

5.4 Final comparison

After the conclusions reached in last plots, a final comparison is done in order to know if it is necessary to calculate the aerodynamic parameters of the kite at every angle of attack or simply use the aerodynamic coefficients for a particular alpha that is in the range of study. For doing that the Lift coefficient with respect to α is plotted for a geometry that is maintained and equal to the one obtained for an angle of attack equal to 5 degrees.

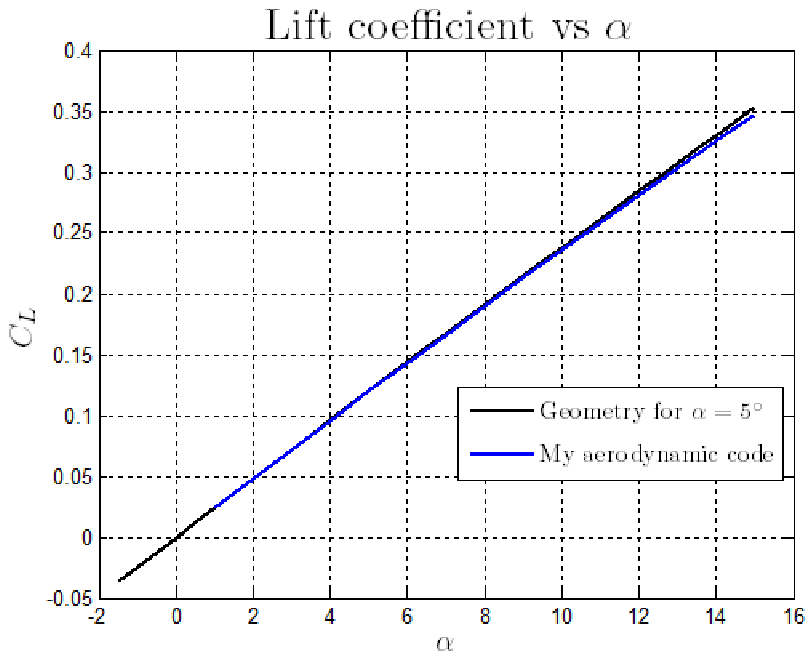


Figure 32: Lift coefficient vs Angle of attack

It can be seen in Figure 32 that the C_{L_α} is almost the same, as expected.

In principle this was not the result expected by the author, and has been analyzing the possible causes of this difference between the model and reality. Finally, it has been concluded that is necessary to put some resistance in the joints of the panels (with torsion springs) to model the rigidity of the structure. Unfortunately, the author has not had time to incorporate this change in the project. The rigidity of the kite can be a factor, but do not rule out that there may be others.

6 Conclusions and Future work

In this final section of the project the main conclusions and recommendations are going to be presented. These conclusions include possible improvements and future work to be done regarding the obtained results. The recommendations also include some tips about how to improve the software.

6.1 Conclusions

It has successfully achieved the proposed of the project, which is to find the position of equilibrium of a kite for a given initial conditions. To do this was not taken into account non-stationary effects nor the rigidity of the kite. As already mentioned, the results are unexpected and the author does not have very clear what is happening. The non insertion of the kite rigidity can be a factor that produce these results, but maybe some other factors are affecting them.

This kind of software will be used at the early stages of the design of the kite, for obtaining a preliminary evaluation of several alternatives to the model considered, wasting the lowest amount of computational time.

As could be observed during validation, the program also can be used as a Vortex Lattice Code for any 3D-wing. The user of the program just has to change the geometry of the wing and the geometry of the lattice. This can be very useful for calculate, as a first approximation, the aerodynamic coefficients.

It should be taken into account that this new way of production of energy is still in the development. It was shown, that these approaches are still in the simulation and small prototypes phases, but are expected to reach the application phase soon due to intensive work, interest and awareness of many research teams and sponsors.

6.2 Future work

Apart from all the work done, many things remain to be implemented in this project.

Structural improvements

- As stated before, in all the process the rigidity of the kite is not taken into account. This rigidity could be included modifying the equation 3.3.23. This equation becomes now:

$$\sum \overrightarrow{M_j} \neq \overrightarrow{0} \quad \rightarrow \quad \sum \overrightarrow{M_j} = K \times \Delta\theta_j \tag{6.2.1}$$

  being $\Delta\theta$ the different of angles between the two bars at the node j and K the constant of the spring.

Aerodynamic improvements

- Side-slip wind. In this project only the angle of attack is considered, but the side-slip angle could be an important parameter because the kite is not like a planar wing. As the kite is in 3D, this angle could vary a lot the forces acting on each of the panels, specially for too large side-slip angles, since as it can be seen in the results section, some panels are practically at 90 degrees with respect to the horizontal axis.

Algorithmic improvements

- Using another method than the Newton-Raphson for the calculation of the Jacobian. As it was stated before, there is not a method for solve a system of non-linear equations that guarantee us the convergence. But there are other methods more complex than the one used, that has a better way to reach a solution.

- Trying to reduce the time it takes for the program to calculate everything.

- Accomplishment of a user's graphical interface (GUI) for the program in order to obtain an easier and more comfortable way of interaction of the designer with the software.

Other improvements

- Implementing non-stationary effects. For example, some gusts along the normal flying of the kite.
- Using different types of airfoils with camber. In this project only a symmetric airfoil without camber is used. In future works this software could be implemented in order to introduce many types of airfoils including, for example, Leading edge inflatable kites.

Appendices

Appendix A. — Appendix: Selection of Control Point/Vortex Location

In order to use the horseshoe vortex to represent a lifting surface, it is very important to determine exactly where the vortex and the control point are located to satisfy the surface boundary condition. Traditionally, their locations have been determine by comparison with known results.

In this Appendix, two simple cases are going to be used in order to determine exactly their position.

A.0.1 Simplest Approach: A Flat Plate

Consider representing the flow over a flat plate airfoil by a single vortex and control point. It is going to be determined the spacing between the vortex and control point which produces the same lift than the thin airfoil theory. The flat plate is represented in *Figure 33*.

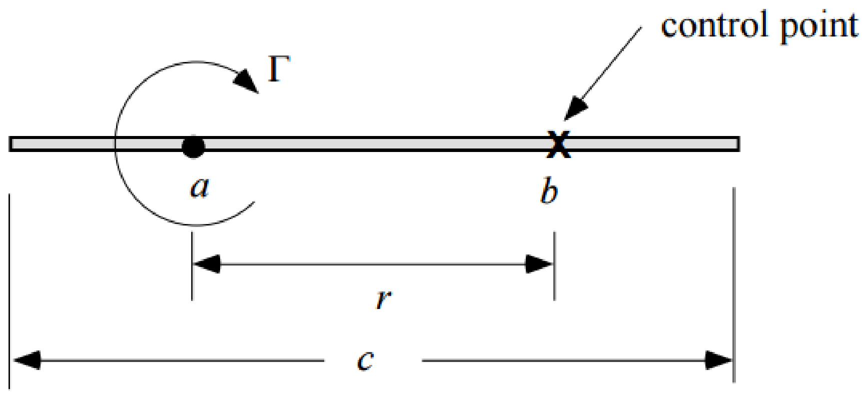


Figure 33: The notation for control point and vortex location analysis [4]

The velocity at the control point, C_p, due to the point vortex is:

$$v_{cp} = -\frac{\Gamma}{2\pi r} \tag{A.0.1}$$

The boundary condition of the flow is equal to:

$$\frac{v_{BC}}{V_\infty} = \frac{df_c}{x} - \alpha \tag{A.0.2}$$

and ignoring camber:

$$v_{BC} = -V_\infty \alpha \tag{A.0.3}$$

Now, we equate v_{BC} and v_{cp}:

$$-\frac{\Gamma}{2\pi r} = -V_\infty \alpha \tag{A.0.4}$$

resulting in the expression for α:

$$\alpha = \frac{\Gamma}{2\pi r V_\infty} \tag{A.0.5}$$

To make use of this relation, recall the Kutta-Joukowsky theorem:

$$L = \rho V_\infty \Gamma \tag{A.0.6}$$

and the result from thin airfoil theory:

$$L = \frac{1}{2}\rho V_\infty^2 c C_l \tag{A.0.7}$$

where $C_l = 2\pi\alpha$.

Equate the expressions for lift, Eqns. A.0.6 and A.0.7 and substitute for α using the expression in Eqn. A.0.5 given above:

$$r = \frac{c}{2} \tag{A.0.8}$$

This defines the relation between the vortex placement and the control point in order for the single vortex model to reproduce the theoretical lift of an airfoil predicted by thin airfoil theory. Since the flat plate has constant (zero) camber everywhere, this case doesn't pin down placement (distance of the vortex from the leading edge) completely. Intuitively, the vortex should be located at the quarter-chord point since that is the location of the aerodynamic center of a thin flat plate airfoil. The next example is used to determine the placement of the vortex.

A.0.2 Determine placement of the vortex using parabolic camber model

Rewrite the velocity at the control point due to the point vortex in a little more detail, where a denotes the location of the vortex, and b the location of the control point:

$$v_{cp} = -\frac{\Gamma}{2\pi(a-b)} \tag{A.0.9}$$

as the boundary condition remains the same, we can equate both equation:

$$-\frac{\Gamma}{2\pi(a-b)V_\infty} = \frac{df_c}{x} - \alpha \tag{A.0.10}$$

Now a parabolic camber is going to be used:

$$f_c(x) = 4\delta\left(\frac{x}{c}\right)(c-x) \tag{A.0.11}$$

substituting in Eqn. A.0.10:

$$-\frac{\Gamma}{2\pi(a-b)V_\infty} = 4\delta\left[1-2\left(\frac{x}{c}\right)\right]-\alpha \tag{A.0.12}$$

Now use the result from thin airfoil theory:

$$L = \frac{1}{2}\rho V_\infty^2 c 2\pi(\alpha+2\delta) \tag{A.0.13}$$

and substitute for the lift from the Kutta-Joukowsky theorem. Obtaining an expression for the circulation of the vortex in terms of the angle of attack and camber:

$$\Gamma = \pi V_\infty c(\alpha+2\delta) \tag{A.0.14}$$

Substitute for Γ from Eqn. A.0.14 into Eqn. A.0.12, and satisfy the boundary condition at x = b:

$$\frac{-\pi V_\infty c(\alpha+2\delta)}{2\pi(b-a)V_\infty} = 4\delta\left[1-2\left(\frac{b}{c}\right)\right]-\alpha \tag{A.0.15}$$

To be true for arbitrary α, δ, the coefficients must be equal:

$$-\frac{1}{2}\left(\frac{c}{b-a}\right) = -1 \tag{A.0.16}$$

$$-\left(\frac{c}{b-a}\right) = 4\left[1-2\left(\frac{b}{c}\right)\right] \tag{A.0.17}$$

Solving the first equation give us the result:

$$b-a = \frac{c}{2} \tag{A.0.18}$$

obtaining the same result obtained above, validating our previous analysis ($r = c/2$).

Now, after solve the second equation and substituting the Eqn. A.0.18, this result appears:

$$\frac{b}{c} = \frac{3}{4} \tag{A.0.19}$$

$$\frac{a}{c} = \frac{1}{4} \tag{A.0.20}$$

Thus the vortex is located at the 1/4 chord point, and the control point is located at the 3/4 chord point. Naturally, this is known as the "1/4 - 3/4 rule." It's not a theoretical law, simply a placement that works well and has become a rule of thumb. Mathematical derivations of more precise vortex/control point locations are available, but the 1/4 - 3/4 rule is widely used, and has proven to be sufficiently accurate in practice. More information in [4].

Appendix B. — Appendix: Budget

In this section it is going to be analyzed an approximation of the budget that is going to be required for the accomplishment of the project.

It should be keep in mind that this kind of software is going to be very useful for the companies due to their low initial investment required and the first insight that the program can give us with a very reduced computational time and effort.

The Code described in this bachelor's thesis has been developed using Matlab ®, as the platform in which it is based. The costs for the acquisition of an individual license are published officially by the developers to be 2000 €, if the software is desired to be handled by several users in a company, the price of this software will be noticeably increased.

The work of the engineer for software development is estimated to be about 300 hours in a period of four months.

- Worker Salary → 8260€
  - Basic Salary → 20€/*hour* → 6000€
  - Transport → 7€/*day* → 560€
  - Social Security [11] (28.3%) → 1700€
- Equipment & Facilities → 2380€
  - Basic Facility Costs [12] → 300€
  - Equipment Maintenance → 80€
  - Software License → 2000€
- Bibliography → 205€
  - Airborne Wind Energy [5]→ 205€
- Total Costs → 10845€

Table 5: Project Budget

Appendix C. — Appendix: Codes

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                           %
%                                MAIN CODE                                  %
%                                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
clear all
close all
%%%%%%%%%%%%%%%%%%%%%%%%%%% Input Paremeters %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Npanels = 4; % Even numbers
AR = 3.866666666666667; % Aspect Ratio
c = 1.5; % [m] Chord of the kite
h = 750; % [m] Height of the kite with respect to see level
Minf = 0; % Mach number
Uinf = 14; % Wind speed
alpha = 5*pi/180; % Angle of attack
l(1) = 100; % [m] Length L0
maxiter = 100000; % Maximum number of iterations
tolL2 = 1e-1; % Tolerance of the error L2
tolInf = 1e-1; % Tolerance of the error Inf
DeltaTheta = 0.0005; %Angle variation for the Jacobian calculation
gamma=0.0001; %Parameter used to control the iterative process
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
isa = ISA(h);
rho = isa.Rhoh;

nx = 2;
ny = Npanels+1;

b0 = AR*c;
b = b0*sqrt(1-Minf^2); % [m] Kite span
```

```
l(2) = b/(ny-1); %Bar length

R = l(1);
[ theta ] = Circunferencia( b,ny,R ); % First approximation to theta
N = numel(theta);

iter=0;
while iter<maxiter
  [ M, L_each, deltay, deltaz, theta, l, Cl_total_total, Cd_total_total,
    yyc, zzc, Ly, Lz ] = Calculo1( theta, AR, c, rho, Minf, nx, ny, Uinf,
    alpha, l, b );

  %%% Finite diferences forward
  for i=1:N
    if i<=N/2
      delta=DeltaTheta;
    else
      delta=-DeltaTheta;
    end
    thetamod=theta;
    thetamod(i)=theta(i)+delta;
    [ M1 ] = Calculo2( thetamod, L_each, deltay, deltaz, l );
    Jacobiano(:,i)=(M1-M)/delta;
  end

  %%% Analytical part of the Jacobian
  for i=1:N
    Jacobiano(1,i)=-sin(theta(i))*l(i);
    Jacobiano(N,i)=cos(theta(i))*l(i);
  end

  Condicion(iter+1)=cond(Jacobiano);
  Rango(iter+1)=rank(Jacobiano);

  deltaTheta=Jacobiano\M;
```

```
thetaNew=theta-deltaTheta;

A=0;
for i=1:N
  A=A+(abs(thetaNew(i)-theta(i))/abs(theta(i)));
end

Error2(iter+1)=sqrt(A/N); % Error L2
ErrorInf(:,iter+1)=(abs(thetaNew-theta)./abs(theta)); %Error Inf

theta=gamma*thetaNew+(1-gamma)*theta;

h=max(z(2:1:end-1))-min(z(2:1:end-1)); %Parameter h
s=max(y)-min(y); %Parameter s

iter=iter+1;

if Error2(iter)<tolL2 && max(ErrorInf(:,iter))<tolInf
   break
end

end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                                %
%                                   FUNTION 1                                    %
%                                                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ M, L_each, deltay, deltaz, theta, l, Cl_total_total,
    Cd_total_total, yyc, zzc, Ly, Lz ] = Calculo1( theta, AR, c, rho, Minf,
    nx, ny, Uinf, alpha, l, b )

for j=1:ny
```

```
    for  i=1:nx
        xLE(j)=0;
        xx(j,:)=linspace(xLE(j),c,nx);
    end
end
n=ceil(ny/2);
A=0;
for  i=2:n
    A=A-cos(theta(i))*l(2);
end
yy(1,1)=A;
yy(1,2)=A;

for  j=2:ny
    for  i=1:nx
        yy(j,i)=yy(j-1,i)+l(2)*cos(theta(j));
    end
end

zz(1)=0;
for  j=2:numel(theta)-1
    for  i=1:nx
        zz(j,i)=zz(j-1)+sin(theta(j))*l(2);
    end
end

S=0;
for  j=1:ny-1
    for  i=1:nx-1
        x1=xx(j,i);
        x2=xx(j+1,i);
        x3=xx(j+1,i+1);
        x4=xx(j,i+1);

        y1=yy(j,i);
```

```
        y2=yy(j+1,i);
        y3=yy(j+1,i+1);
        y4=yy(j,i+1);

        z1=zz(j,i);
        z2=zz(j+1,i);
        z3=zz(j+1,i+1);
        z4=zz(j,i+1);

        p1=[x1 y1 z1];
        p2=[x2 y2 z2];
        p3=[x3 y3 z3];

        normal(j,:) = cross(p1 - p2, p1 - p3);

        xc=[x1 x2 x3 x4];
        yc=[y1 y2 y3 y4];
        zc=[z1 z2 z3 z4];

        geom=polygeom(xc,yc);
        xxc(j,i)=3/4*c;
        yyc(j,i)=geom(3);
        zzc(j,i)=(z2+z1)/2;

        S=b^2/AR;
        S_each(i,j)=l(2)*c;
    end
end

for i=1:numel(zzc)
    deltay(i)=yy(i+1,1)-yy(i,1);
    deltaz(i)=zz(i+1,1)-zz(i,1);
end
%%
N=length(xxc(:));
```

```
for j=1:length(yyc(:,1))
  xLEm(j)=0;
end
xLEm=reshape(xLEm,[numel(xLEm) 1]);

m=0;
for k=1:N
  x1=100*b;
  x4=x1;
  x2=1/4*c;
  x3=x2;

  y1=yyc(k)-deltay(k)/2;
  y2=y1;
  y3=yyc(k)+deltay(k)/2;
  y4=y3;

  z1=zz(k);
  z2=z1;
  z3=zz(k+1);
  z4=z3;

for p=1:N
  xp=xxc(p);
  yp=yyc(p);
  zp=zzc(p);

  Cpk(p,k)=horseshoe(1,x1,x2,x3,x4,y1,y2,y3,y4,z1,z2,z3,z4,xp,yp,zp);
  CpkMod(p,k)=horseshoeMod(1,x1,x2,x3,x4,y1,y2,y3,y4,z1,z2,z3,z4,xp,yp,zp);
  end
end

Cpkk=sum(transpose(Cpk));

for i=1:numel(alpha)
```

```
  n1=[cos(alpha(i)),0,sin(alpha(i))];
  for j=1:N
    n2=normal(j,:);
    cosine=-(dot(n2,n1)/(norm(n1)*norm(n2)));
    alpha_each(i,j)=pi/2-acos(cosine);
  end
end

for i=1:numel(alpha)
  for j=1:N
    w=Uinf*(-alpha_each(i,j));
    Capgamma(j)=w/Cpkk(j);
    Cl_lifting_inc_each(j,i) = Capgamma(j)*rho*Uinf*deltay(j)/(0.5*rho*Uinf
   ^2*S_each(j));
    Cl_lifting_com_each(j,i) = Cl_lifting_inc_each(j,i)/sqrt(1-Minf^2);
    L_each(j,i)=Cl_lifting_com_each(j,i)*(0.5*rho*Uinf^2*S_each(j));
    L_total(j,i)=L_each(j,i)*cos(theta(j+1))*cos(alpha(i));
    Lz(j,i)=L_each(j,i)*cos(theta(j+1))*cos(alpha_each(i,j));
    Ly(j,i)=-L_each(j,i)*sin(theta(j+1))*cos(alpha_each(i,j));
  end
  W(:,i) = CpkMod*Capgamma';
  D(i)=0;

  for j=1:N
    D_each(j,i)=-rho*Capgamma(j)*W(j,i)*deltay(j);
  end
end

L_total_total=sum(L_total);
Cl_total_total=L_total_total/(0.5*rho*Uinf^2*S);

D_total=sum(D_each);
Cd_total_total=D_total/(0.5*rho*Uinf^2*S);

%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Ry=0;
Rz=0;

N=numel(theta);
for i=1:N-2
   Ry=Ry+L_each(i)*sin(theta(i+1));
   Rz=Rz-L_each(i)*cos(theta(i+1));
end
%%%%
l(N)=l(1);
A=0;

for i=2:N-1
   l(i)=l(2);
   A=A+l(i)*cos(theta(i));
end

theta(1)=acos(-A/(2*l(1)));
theta(N)=-theta(1);
%%%%
T(1)=(cos(abs(theta(N)))*(-Rz)-sin(abs(theta(N)))*Ry)/(sin(theta(1)+abs(
     theta(N))));
T(N)=(cos(theta(1))*(-Rz)+sin(theta(1))*Ry)/(sin(theta(1)+abs(theta(N))));
%%%%
for i=1:N-2
   Fy(i)=-L_each(i)*sin(theta(i+1));
   Fz(i)=L_each(i)*cos(theta(i+1));
end

Ty(1)=-T(1)*cos(theta(1));
Ty(2)=-(Ty(1)+Fy(1));

Tz(1)=-T(1)*sin(theta(1));
Tz(2)=-(Tz(1)+Fz(1));
```

```
for j=2:N-2
   Ty(j+1)=-(-Ty(j)+Fy(j));
   Tz(j+1)=-(-Tz(j)+Fz(j));
end
for j=1:N-2
   To1=[0,-Ty(j),-Tz(j)];
   To2=[0,Ty(j+1),Tz(j+1)];
   rTo1=[0,-deltay(j)/2,-deltaz(j)/2];
   rTo2=[0,deltay(j)/2,deltaz(j)/2];

   if j==1
      To1=[0,Ty(j),Tz(j)];
   end

   MTo1=cross(rTo1,To1);
   MTo2=cross(rTo2,To2);

   M(j+1)=MTo1(1)+MTo2(1);
end

M(1)=0;
M(N)=0;
for i=1:N
   M(1)=M(1)+l(i)*cos(theta(i));
   M(N)=M(N)+l(i)*sin(theta(i));
end

M=M';
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
%                                FUNTION 2                                     %
```

```
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ M1 ] = Calculo2( thetamod, L_each, deltay, deltaz, l )
Ry=0;
Rz=0;
theta=thetamod;
N=numel(theta);

for i=1:N-2
   Ry=Ry+L_each(i)*sin(theta(i+1));
   Rz=Rz-L_each(i)*cos(theta(i+1));
end
%%%
T(1)=(cos(abs(theta(N)))*(-Rz)-sin(abs(theta(N)))*Ry)/(sin(theta(1)+abs(
    theta(N))));
T(N)=(cos(theta(1))*(-Rz)+sin(theta(1))*Ry)/(sin(theta(1)+abs(theta(N))));
%%%
for i=1:N-2
   Fy(i)=-L_each(i)*sin(theta(i+1));
   Fz(i)=L_each(i)*cos(theta(i+1));
end

Ty(1)=-T(1)*cos(theta(1));
Ty(2)=-(Ty(1)+Fy(1));

Tz(1)=-T(1)*sin(theta(1));
Tz(2)=-(Tz(1)+Fz(1));

for j=2:N-2
   Ty(j+1)=-(-Ty(j)+Fy(j));
   Tz(j+1)=-(-Tz(j)+Fz(j));
end
for j=1:N-2
   To1=[0,-Ty(j),-Tz(j)];
   To2=[0,Ty(j+1),Tz(j+1)];
```

```
    rTo1=[0,-deltay(j)/2,-deltaz(j)/2];
    rTo2=[0,deltay(j)/2,deltaz(j)/2];

    if j==1
        To1=[0,Ty(j),Tz(j)];
    end

    MTo1=cross(rTo1,To1);
    MTo2=cross(rTo2,To2);

    M1(j+1)=MTo1(1)+MTo2(1);
end

M1(1)=0;
M1(N)=0;
for i=1:N
    M1(1)=M1(1)+l(i)*cos(theta(i));
    M1(N)=M1(N)+l(i)*sin(theta(i));
end

M1=M1';
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
%                              HORSESHOE CODE                                  %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ w ] = horseshoe( gamma,x1,x2,x3,x4,y1,y2,y3,y4,z1,z2,z3,z4,xp,yp
    ,zp )
[u1 v1 w1]=vline(gamma,x1,y1,z1,x2,y2,z2,xp,yp,zp);
[u2 v2 w2]=vline(gamma,x2,y2,z2,x3,y3,z3,xp,yp,zp);
[u3 v3 w3]=vline(gamma,x3,y3,z3,x4,y4,z4,xp,yp,zp);
```

```
w=w1+w2+w3;
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                      %
%                              VLINE CODE                              %
%                                                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ u v w ] = vline(gamma,x1,y1,z1,x2,y2,z2,xp,yp,zp)
eps=0.001;

r1_r2_x = (yp-y1)*(zp-z2)-(zp-z1)*(yp-y2);
r1_r2_y = -(xp-x1)*(zp-z2)+(zp-z1)*(xp-x2);
r1_r2_z = (xp-x1)*(yp-y2)-(yp-y1)*(xp-x2);
r1_r2 = r1_r2_x + r1_r2_y + r1_r2_z;

modsq_r1_r2=(r1_r2_x)^2+(r1_r2_y)^2+(r1_r2_z)^2;

modr1 = sqrt((xp-x1)^2+(yp-y1)^2+(zp-z1)^2);
modr2 = sqrt((xp-x2)^2+(yp-y2)^2+(zp-z2)^2);

if modr1<eps
  u=0;
  v=0;
  w=0;
  elseif modr2<eps
  u=0;
  v=0;
  w=0;
  elseif modsq_r1_r2<eps
  u=0;
  v=0;
  w=0;
end
```

```
r0_r1 = (x2-x1)*(xp-x1)+(y2-y1)*(yp-y1)+(z2-z1)*(zp-z1);
r0_r2 = (x2-x1)*(xp-x2)+(y2-y1)*(yp-y2)+(z2-z1)*(zp-z2);

K=gamma/(4*pi)/modsq_r1_r2*(r0_r1/modr1-r0_r2/modr2);

u=K*r1_r2_x;
v=K*r1_r2_y;
w=K*r1_r2_z;
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                         %
%                              ISA CODE                                   %
%                                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [c]=ISA(h)
Tsl=288.16;
Psl=101325;
Rhosl=1.225;
gsl=9.81;
aLr=-0.0065;
Th=Tsl+aLr*h;
a0=340.29;
theta=Th/Tsl;

if h<11000
  theta=1-(2.2557e-5)*h;
  Th=theta*Tsl;
  Ph=Psl*theta^5.2621;
  delta=Ph/Psl;
  Rhoh=Rhosl*theta^4.2621;
  sigma=Rhoh/Rhosl;
  a=a0*sqrt(theta);
```

```
    c=struct('theta',theta,'Th',Th,'Ph',Ph,'delta',delta,'Rhoh',Rhoh,'sigma',
      sigma,'a',a);
else
    theta=0.7519;
    Th=theta*Tsl;
    Ph=Psl*0.223*exp(-(0.0001578)*(h-11000));
    delta=Ph/Psl;
    Rhoh=Rhosl*0.297*exp(-(0.0001578)*(h-11000));
    sigma=Rhoh/Rhosl;
    a=a0*sqrt(theta);

    c=struct('theta',theta,'Th',Th,'Ph',Ph,'delta',delta,'Rhoh',Rhoh,'sigma',
      sigma,'a',a);
    end
end
```

References

[1] Danish association of wind industry web site. www.windpower.org.

[2] How does a kite surf fly? http://www.kitegeneration.com/kite-surf-fly/.

[3] Newton-raphson method for nonlinear systems of equations. http://www.aip.de/groups/soe/local/numres/bookcpdf/c9-6.pdf.

[4] Aerospace and O. E. . V. Tech. Aerodynamics of 3d lifting surfaces through vortex lattice methods. http://www.dept.aoe.vt.edu/.

[5] U. Ahrens, M. Diehl, and R. Schmehl. *Airborne wind energy*. Springer Science & Business Media, 2013.

[6] J. Alonso-Pardo and G. Sánchez-Arriaga. Kite model with bridle control for wind-power generation. *Journal of Aircraft*, 2015.

[7] J. D. Anderson Jr. *Fundamentals of aerodynamics*. Tata McGraw-Hill Education, 1985.

[8] C. L. Archer and K. Caldeira. Global assessment of high-altitude wind power. *Energies*, 2(2):307–319, 2009.

[9] J. J. Bertin and M. L. Smith. Aerodynamics for engineers. 1998.

[10] M. Canale, L. Fagiano, and M. Milanese. Power kites for wind energy generation. *IEEE Control Systems Magazine*, 27(6):25–38, 2007.

[11] Government of Spain. Contribution bases and rates. http://www.seg-social.es.

[12] J. Hadzima. How much does an employee cost? *Boston Bussines Journal*.

[13] J. Katz and A. Plotkin. *Low-speed aerodynamics*, volume 13. Cambridge University Press, 2001.

[14] G. M. Maneia. Aerodynamic study of airfoils and wings for power kites applications. *Politecnico di Torino, Italy-Master's thesis*, 93:94–95, 2007.

[15] W. H. Mason. *Applied Computational Aerodynamics: A Modern Engineering Approach.* Cambridge Aerospace Series, 1998.

[16] T. Melin. A vortex lattice matlab implementation for linear aerodynamic wing applications. *Master's Thesis, Department of Aeronautics, Royal Institute of Technology (KTH), Stockholm, Sweden*, 2000.

[17] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. Numerical recipes in c: the art of scientific computing. *Cambridge University Press, Cambridge, MA,*, 131:243–262, 1992.