

University Degree in Aerospace engineering
Academic Year (e.g. 2018-2019)

Bachelor Thesis

“Real Time Control For Airborne Wind Energy Simulators”

Alberto José García Muñoz
Gonzalo Sánchez Arriaga
Leganés, June 10th, 2019



This work is licensed under Creative Commons **Attribution – Non Commercial – Non Derivatives**

ABSTRACT

This thesis develop a real time simulator for an acrobatic kite, called LAKSA Simulink. The simulator is base on a Matlab code, called LAKSA, which describe the kite behavior using Lagrangian equation. This code is made of functions with different tasks, from describing the kite behavior to plot it. This thesis implement Matlab code into Simulink and add some functionalities: user input, allowing the user to control the kite; and real time simulation. This thesis aim to create a simulator that work as efficient as possible, that is why during the development of this thesis, the improvement of the performance of the simulator and reduction of the computational time have been taken into consideration. During this thesis, different data regarding the kite behavior and the simulation performance are obtained and they are analyzed in order to study the effect of the user input in the kite behavior, and the influence of each task of the simulation in the simulator performance.

ACKNOWLEDGE

I want to thank all my friends, the ones from Badajoz, who are still there even in the distance, and all the friends I made during these four years in Madrid and in Stuttgart, who has made this trip with me. I want to thank my family for all the support, and for making possible live out of my city for four years. Last but not least, I want to thank especially my tutor, Gonzalo Sánchez Arriaga, for helping me in the development of this thesis and all the useful advice, which have made me improve as an engineering.

Contents

1	Introduction	3
1.1	Project motivation	3
1.2	Thesis goal	4
2	State of art	5
2.1	LAKSA simulator	5
2.2	Actual state of the AWE industry	7
3	Two-line kite model	14
3.1	Lagrangian function and forces	17
3.2	Equation of motion	18
3.3	Limits of the model	19
4	Simulink program	21
4.1	Recreate LAKSA code in Simulink	21
4.2	User input	24
4.3	Verification of Simulink calculation	28
4.4	Real time simulation	30
4.5	Kite display	32
4.6	Physical consistency and influence of the control law	37
4.7	Task influence in real time synchronization	41
5	Regulatory Framework	47
6	Budget	48
7	Conclusion	49
8	Apendix A: LAKSA program and function	
9	Apendix B: Simulink Blocks	

List of Figures

1	Makani M600 system [2]	4
2	Deep Water Concept from Makani [2]	4
3	Kite Display from LAKSA program [8]	7
4	AWEC logo [3]	7
5	Example of optimized AWE flight trajectory [4]	9
6	Vortex model, Delf University [4]	10
7	Wind influence in tether force [4]	10
8	Race For Water Yacht, SkySail © [4]	11
9	Drive Shaft, Munich School of Engineering [4]	12
10	Windswept and Interesting Ltd. concept [4]	12
11	University of Strathclyde concept, rotatory system [4]	13
12	Kite coordinate systems [1]	15
13	Change from S_2 to S_B [1]	16
14	Simulink model [9]	22
15	Solver options [9]	23
16	Joystick V1.1 - δ [9]	25
17	Joystick V1.2 - $\dot{\delta}$ [9]	25
18	Joystick V1.3 - $\ddot{\delta}$ [9]	26
19	Simulink final model [9]	26
20	First model with δ as input [9]	27
21	Final control model with δ as input [9]	27
22	Controller final model [9]	28
23	Verification process	29
24	Example of verification process [10]	30
25	Mixed Ticks plot, Simulink graph (x-axis: time (s), y-axis (missed ticks (-)) [9]	31
27	Comparison of δ values [10]	32
26	Simulink model, with dimensional signal [9]	32
28	Comparison of state vector values [10]	33
29	Simulink final model [9]	35
30	LAKSA Display [8]	36
31	Simulink Display [9]	36
32	Simulink Window set up [9]	37
33	Kite physic plot [11]	38
34	Case 1 [11]	39
35	Case 2 [11]	39
36	Case 3 [11]	40
37	Case 4 [11]	40
38	Plot rate t=1s	42
39	Plot rate t=5s	42
40	Plot rate t=0.5s	43
41	Figure not initialized [10]	43
42	Figure initialized [10]	44
43	Case t=1s	45
44	Case t=5s	45
45	Case t=0.1s	46
46	Case t=0.4s	46
47	PD struct [8], [9]	

LIST OF TABLES

48	LAKSA kite display [8]
49	Interpreted Matlab Fcn [9]
50	Mux and demux [9]
51	Knob connected to a constant [9]
52	Integral block [9]
53	Rate transition Block [9]
54	Terminator [9]
55	To file box [9]

List of Tables

1	Dimensionless parameters used in the simulation [12]	18
2	Kite parameters [12]	21
3	Maximum values configuration	38
4	Employees cost	48
5	Tool prices	48
6	Total cost	48
7	PD Kite [12]	
8	PD Aero [12]	
9	PD Env [12]	
10	PD Tether [12]	
11	Dimensionless parameters used in the simulation [12]	
12	Kitematic component	
13	Kite forces and moments	
14	Other components	

1 Introduction

1.1 Project motivation

This thesis is part of a bigger project consisting in using kites for engineering purposes, such as generate electricity or pulling cargo ships. A lot of companies and universities all around the globe are working on this topic, focus in different concept and in different steps of the development process. In the actual situation of experiment climate change, different concepts regarding the generation of renewable electrical energy have been faced. Renewable energy can be obtained from different sources, like wing or sun, which allows to obtain energy in a sustainable way, and avoid the harmful of the environmental. This technology can be include inside the category of Airborne Wind Energy (AWE), which is a technology aiming to get renewable energy from the wind, using kites, and thus reducing the use of fossil fuels and pollution. Nowadays, only 4% of the total electricity is generated from Wind Energy [5], but this figure can improve significantly due to the potential of this technology. The main advantage of this system is that wind energy can be obtained from places, like high altitudes or offshore stations, where conventional turbines can not be used. Some experiments show that this technology can be feasible, but more investigations must be carried on to see if it is profitable, from a global perspective. One of the main challenges of this technology is to proof that they are worth in the long term, which imply low operational and maintenance cost; so the initial investment is justified. Simulators, like the one developed by this thesis, are very useful in order to test different concepts at a low cost and solve challenges that must be facing in the real world.

Regarding the use of kite to generate energy, different concepts have been facing, although the basic behind of all of them is to obtain energy from the flight of the kite in crosswind conditions. This energy can be used for different purposes: f.e. energy can be generated from turbines attached to the kite, and with a generator in the ground, energy can be obtained from the traction forces of the tethers. One advantage of this system is that the turbines will operate at higher altitudes than conventional wind turbines. In offshore cases, they can operate at distances more far away from the coast than conventional turbines. One advantage is that they are lighter that conventional turbines, reducing costs and they have smaller impact on the landscape. Another use of this energy is pulling cargo boats, reducing the pollution of the engines. One example of the potential of this technology is found in the company Makani and its M600 system. Its concept is shown in figure 1. With the M600 system Makani company aims to get energy from offshore places, where conventional turbines can not be used. Its system is 90% lighter than conventional turbines [2], allowing them to use them in deep water offshore, taking advantage of zones nowadays unused. In addition, with a proper technology, they can be cheaper than conventional turbines while generating the same amount of energy. Deep water concept can be observed in figure 2



Figure 1: Makani M600 system [2]

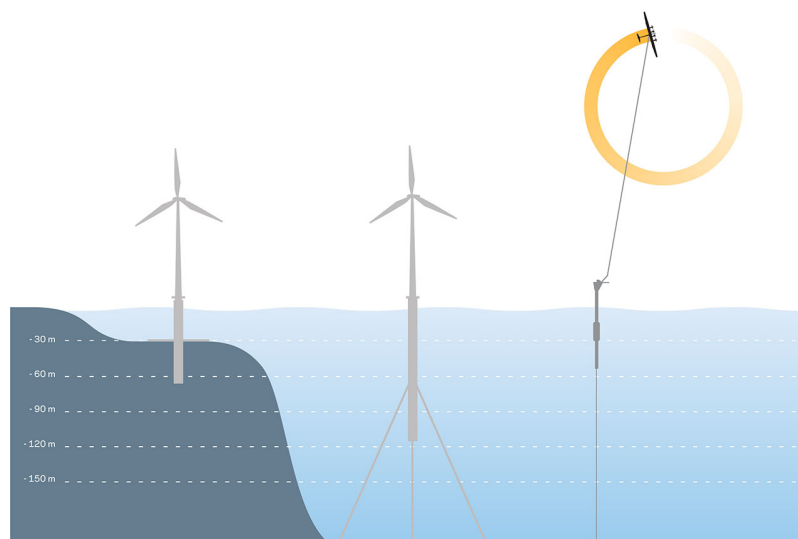


Figure 2: Deep Water Concept from Makani [2]

1.2 Thesis goal

The goal of this thesis is the development of a kite simulator in Simulink which allows the user to control the kite in real time. The essential of the simulator is a set of Matlab function, obtaining from the simulator called LAKSA (Lagrangian Kite Simulator) [8], which are implemented in Simulink. LAKSA simulator is a Matlab code developed by Universidad Carlos III de Madrid, which describe the kite behavior using Lagrangian equations. This thesis add functionalities to LAKSA simulator, like real time simulation and user input. Simulink is the tool chosen because LAKSA code can be easily implemented and

it allows to perform the simulation at real time, among others tasks. The functionalities added by this thesis to the LAKSA simulator are:

- Simulation at real time: simulate the kite behavior at real time, including kite control and display.
- Kite display: In this thesis, the kite is display at the same time as the simulation is performed, rather than in LAKSA program, where the kite is plotted after the simulation, when the state vector is calculated for all the t span. This part is the most computational time demanding of the whole program. During the development process, some modifications are done to LAKSA functions in order to improve simulation efficiency.
- User input: The user can control the kite using a knob, which has influence in the control law, rather than in the original code where the user can only influence the control law of the kite with a control law function, limiting the possibilities of the model.

The final goal of the project is to be able to control the kite autonomously using a machine, which will control the tether length, but previously, it is needed to know the exact behaviour of the kite in different situations and its response to different inputs. At this point, a simulator is the best solution, as it will allow to test different models and different control laws with a very low cost. As consequence of good engineering practices, in this thesis, a Simulink program is created in such a way that it has a modular behavior, and it is properly structured; so it can be easily improved in future researches and the result of other investigation can be applicable, f.e. a better aerodynamic model or a trajectory optimization. Simulink program is created in such a way that the original Matlab code functions are used as much as possible, as they reproduce faithfully the kite physic, with very precise calculations. In addition, the structure of the code is very clever and must be conserved as much as possible. It must be noticed that one important factor that affect the real time simulation is the computational time, and that is why the Simulink program must be done in an efficient way, avoiding all unnecessarily computations.

2 State of art

At the time to face this thesis, to understand the state and the requirement of the sector, two sources have been used: papers of the investigations carried on by professor Gonzalo Sanchez Arriaga in UC3M, in collaboration with another universities, and the actual state of the kite industry. To know the actual state of kite industry the best reference is found in Airbone Wind Energy Conference (AWEC) 2017 (it is the most updated, as there were no conference in 2018), where all the player of this sector take part.

2.1 LAKSA simulator

The investigations carried on in University Carlos III de Madrid regarding kites, driven by professor Gonzalo Sánchez Arriaga, covers a variety of kite topics, from stability, to trajectories or dynamic modelling. Thanks to the papers: "Kite dynamic modelling" [1] and "LAKSA (LAgrangian Kite SimulAtor) project" [8], the dynamic behind the kite and its physic can be understand; and how Matlab kite simulator, called LAKSA, works. It is

a set of matlab equations developed by Gonzalo Sanchez-Arriaga and Alejandro Pastor-Rodriguez. Matlab code is done in an universal way, with the advantage that it can be easily applicable to different kites. In addition, it is able to simulate with different wind patterns and the physic, aerodynamic and control model can be easily implemented, in case that better models are developed. The program developed during this thesis will be strongly dependant on LAKSA functions. The dynamic analysis is important, due to the fact that the control law will have an important effect on it.

LAKSA simulator is a software tool, created using matlab. Its goal is to perform simulations of a kite or fixed-wind drone, considered as rigid body, which is a good advantage of this simulator with respect to the most common ones which consider the device as a single point. Understand the LAKSA simulator is crucial for an efficient development of the Simulink program, as it is based on it [8]. LAKSA simulator has 4 modules:

- Kite Flex
- Kite Acrobat
- Kite Surf
- Kite Elastic

The module use in this thesis is Kite Acrobat, but it is important to say that the 3 first modules (Kite Flex, Kite Acrobat and Kite Surf) are very similar, as all of them simulate with Lagrangian equation, assume inelastic tether, and solve with ordinary differential equation methods. Another important assumption is the consideration of the tether as a straight line, which will simplify the calculations. This assumption can only be applied when the tension is large compared with aerodynamic and gravitational forces of the tether, and it can not be applied during take off and landing. Kite Elastic module consider a tether with an elastic behavior, but as it is out of the scope of this thesis, so it is only going to be named.

LAKSA has as input from the user: initial condition, physical parameter and control law. It is created in such a way that the user can easily modify them, and with the modification of these parameters, different kites can be simulated. The control law is a function, depending on time. The output of LAKSA is the kite state vector, which is a result of the integration of the equation of motions. With the state and control vector, all relevant quantities (dynamical, physical...) can be obtained, and LAKSA plot different simulation result and display the kite at the end of the simulation. Regarding dimensional and non-dimensional simulation, the program is prepared in such a way that the user only interact with dimensional quantities, which help in the physical understanding and it is close to the reality. On the other hand, the calculation are done with non-dimensional quantities, which is appropriate from a numerical point of view. The kite display of the LAKSA program is shown in figure 3.

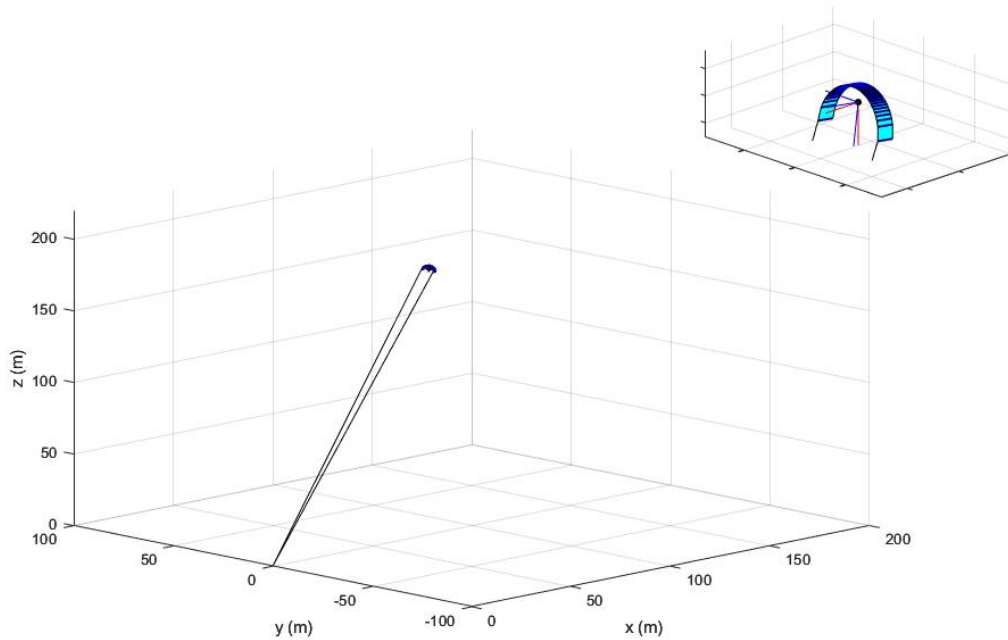


Figure 3: Kite Display from LAKSA program [8]

2.2 Actual state of the AWE industry

The thesis goal is alignment with the goal of the participants in the Airborne Wind Energy Conference (AWEC, figure 4). AWEC documentation allows to know the actual state of the technologies related with Airborne Wind Energy (AWE) and the huge number of companies and institutions all around the world working on this topic, which shows the potential impact that this technology can have in the future [3]. As this topic is nowadays under development, different concepts regarding the use of this technology are considered; some of them are similar to the one of this thesis, another too different, but all must be analyzed to understand the goal of this thesis and its limitations. In addition, this information provides a global vision of this topic, as they are focused in such a different fields.



Figure 4: AWEC logo [3]

Progress and challenges of AWE

Energy is a very important issue of the actual society, due to the fact that the daily demands on energy is increasing, and the way that this energy is obtained has a big impact in the environmental. Past decades the energy has been mainly created using fossil fuels, which has leads the planet to the actual situation of experimentation of climate change. For all these reasons a lot of researches regarding renewal energy are carried on

nowadays. In the other hand, transport industries use carbon fuel for its engines, polluting the atmosphere, but this problem can be solved with this technology, f.e. using kites for pulling boats or obtaining green energy to use in electric vehicles. Some of the challenges that this technology must face in order to be feasible will be explained in detail in further sections, as a summarize they are the following:

- Operational challenge: kites must be able to work during long time services, so they must have high reliability. High maintenance cost will reduce the profit and thus the willingness of invest.
- Environmental challenge: wind has unexpected behaviour, that some times it can not be predicted. In addition the wing has turbulence effect that can be very harmful. Due to high altitudes altitudes considered, these effects are even more important than with conventional turbines and kites must be able to deal with them. They must suffer all kind of environmental conditions (snow, wind...).
- Aviation and ground safety: This problem has not been facing deeply, and as it is a new technology, safety measures from other systems can not be applied.
- Low weight challenged. As in aviation, low weight is always a big concern. Low weight design usually imply low cost and simplicity.
- The interaction of this device with other air user, like aeroplanes or drones must be considered. Other iterations must be taken into account too, like: radar influence, noise emission, visual impact...

Simulation

Simulation is a key feature in the development process, as it allows to know the behaviour of the kite at a very low cost as there is no need of build one to make test. This topic is extremely related with this thesis, as this thesis develop a simulator, so it is very interesting to know the different approachs. Kiteswarms has develop an interesting project: the creation of a Open Source simulator, in order to facilitate the AWE community the use of simulators and the development of them. It is called: "AWE Standardized Open-source Model Environment". Another interesting project is the creation of a program which optimize AWE flight trajectories. It has been done by Ampyx Power B.V. and it consist in the use of a Matlab/Octave toolbox called OpenAWE. This toolbox solve optimal control power in AWE systems minimizing the user cost function and solving the constrain in the solution of the problem [6]. One example can be seen in figure 5.

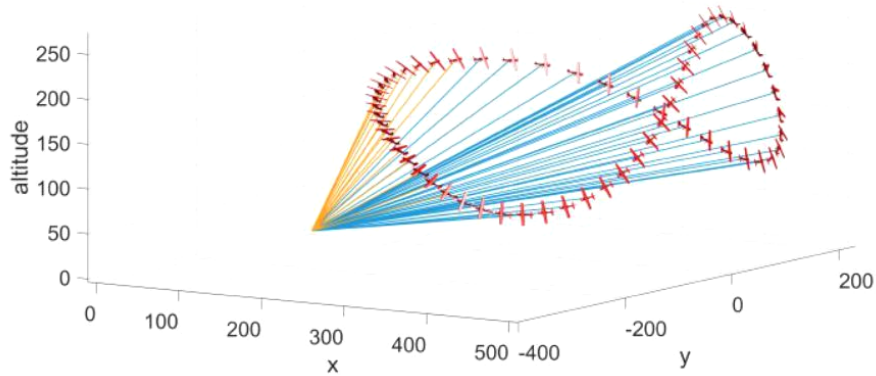


Figure 5: Example of optimized AWE flight trajectory [4]

Aerodynamic and physic of the kite

The aerodynamic of the kite is an important topic, as it is not developed as deep as aircraft aerodynamic. In the kite surfing industry, the kites are designed with empirical methods, due to the absence of alternatives, in particular, due to the absence of numerical methods. Although in this thesis, a very simple model is used, it is worth it to consider the actual state of this topic, as a better aerodynamic model must be implemented in the future in the simulator. A proof of this lack of knowledge is that with a Vortex lattice method, $Cl_\beta = -0.49$ but with the data from para-foil (which must have similar behaviour) $Cl_\beta = -0.037$.

In the Aerospace department of Delf University they are developing a vortex model for time dependent vortex shedding at separation location and trailing edge. Its aim to describe the flow reattachment phenomena on suction and pressure surface in multiple-wake vortex lattice model [4]. The model is shown in figure 6. The french company "Beyond The Sea", is a company focus in using kites for pulling boats. They are trying to optimize the Kite profile using Reynolds-Averaged-Navier-Stockes Flow Simulations. For kites using for pulling cargo boats, 3D lifting line method has been developed to calculate aerodynamic forces and performance of the kite. In the research, different 3D kite profile are analyzed, varying the Maximum Thickness and the Maximum Thickness Position. The main advantage of the simulation is the avoidance of manufacture so many profiles.

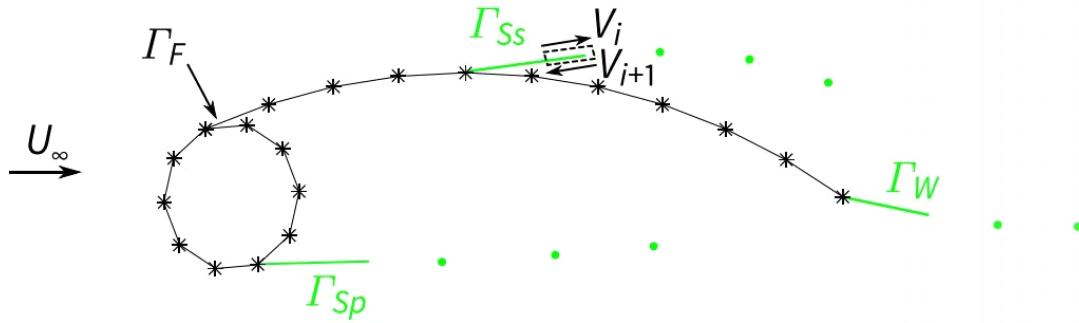


Figure 6: Vortex model, Delf University [4]

On the other hand, the study of the tether forces is necessarily to optimize the cost. Semnan University is investigating the effect of the aerodynamic forces in them. The tether forces must be kept inside a range: a maximum value must not be surpassed to avoid breakages, and a minimum value must not be undertaken, to have always control in the kite. In addition a tether that can support forces bigger than ones they are never going to suffer will be no optimal, as it will increase the weight, reducing the performance and increasing the cost. In figure 7, tether forces for different wind speed are plotted.

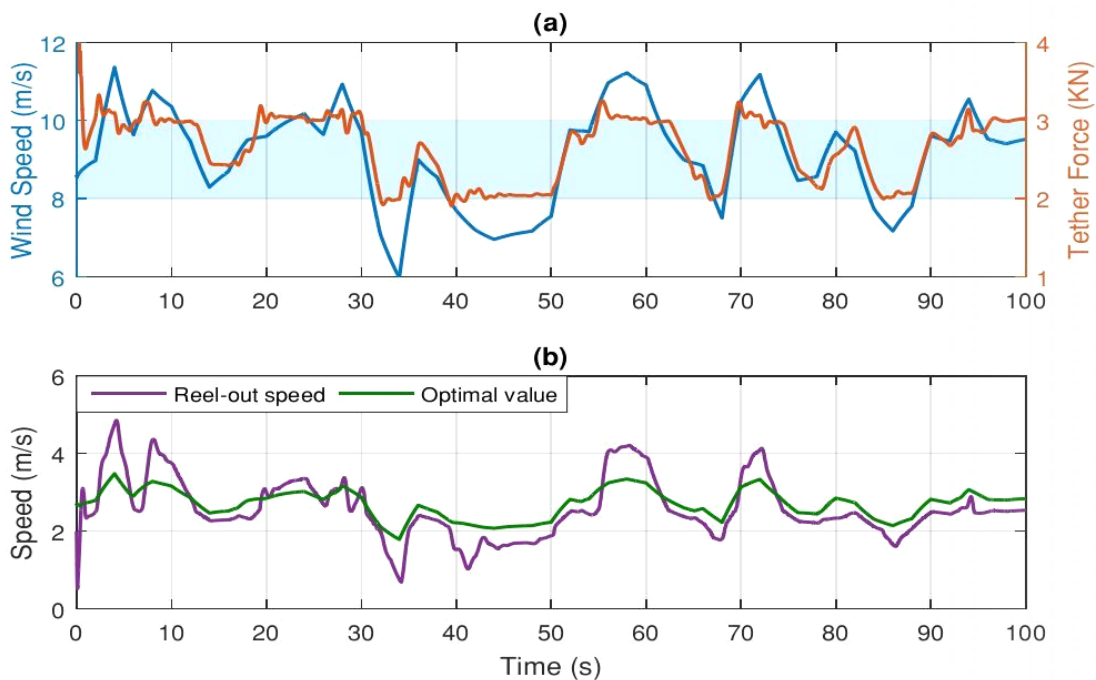


Figure 7: Wind influence in tether force [4]

Real application

In the introduction the Makani project has been explained, as it is the best example of a real application of this technology, because it is the most developed project. In this

section other real applications will be explained. One of the uses of the kites is to pull boats. The company "SkySails" power has developed a model to drive a yacht. They have build a yacht "Race for Water" which has cross the Atlantic using 100% renewable energy, see figure 8. This yacht used solar panel and afterwards it has been equipped with a kite. They have collect data during all the research and they have came up with a very interested conclusion: $30m^2$ of kite can propel the boat as fast as $500m^2$ of solar panels [7].



Figure 8: Race For Water Yacht, SkySail © [4]

Others

In this section the state of some investigation of problems that the AWE technology must face are explained. Although they does not apply directly to this thesis, it is important to know them to have a good overview of the technology.

- Take off and landing:

One mayor problem for facing an autonomously flight is the autonomous take-off and landing. Politecnico di Milano is carrying on a research in this topic, trying to find the most optimum technique. In this maneuver is when the tether suffer the biggest forces, and this is an issue to take care of, as it will condition the tether material, cost, weight..., and kite performance in general.

The Netherlander company "Ampyx Power B.V." has made an interesting approach in this topic which is to use a small plane or a drone to deploy the kite. The small aircraft will deploy the kite and when it is deployed the plane will go back to the base. The energy used for the aircraft is insignificant in comparison with the one obtaining by the kite. This will reduce the tension that the tether must support, reducing the material and the cost of the kite while the energy obtaining from the kite is the same [6].

- Drive shaft:

In the case of a tether connected to the ground and generating energy from to the tension forces, an interesting point of focus is the efficiency of the drive in the ground,

in order to convert as much energy as possible into electricity. Munich School of Engineering is working in a efficient electrical drive, to be used in this approach. In figure 9 it can be observed a real Drive Shaft model. Component I transform the energy to electricity, the kite is attached to component II, and component III is the torque sensor.

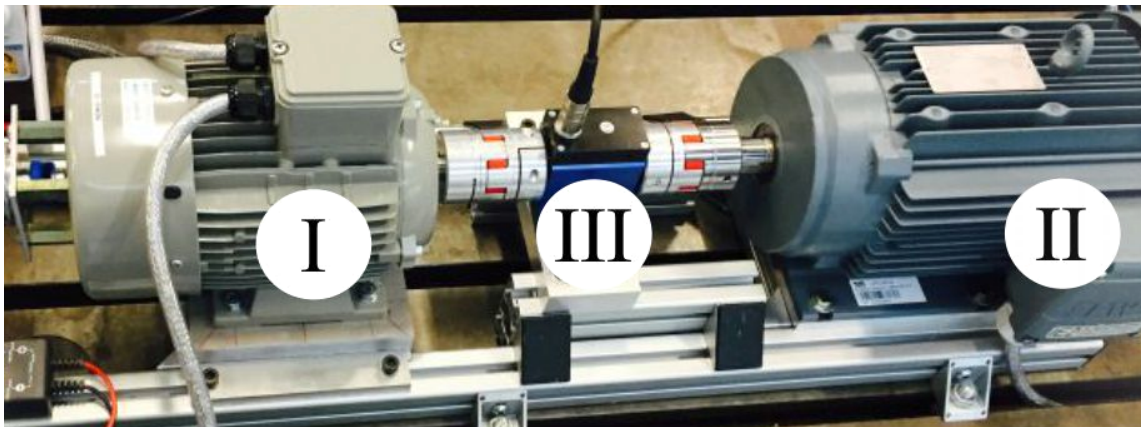


Figure 9: Drive Shaft, Munich School of Engineering [4]

- Kite network:

Some research are investigating the behaviour of a network of kite instead of a single one. It has the advantage that it reduces the cost per kite, but it add some complexity to the system, as the interaction between them must be consider. Windswept and Interesting Ltd., and University of Strathclyde are working in this concepts. Its respective system can be seen in figure 10 and 11 respectively.



Figure 10: Windswept and Interesting Ltd. concept [4]



Figure 11: University of Strathclyde concept, rotatory system [4]

- Regulation, Policy and Safety:

An important topic is the regulation about this new technology. A bit challenge to deal with are secure procedures, the interaction with other air user, interaction with the atmosphere... In addition government must regulated these devices, which will result in constrains to the technology and the development.. One of the goals is to be able to control the kite autonomously. This lead to some problem, the software must be 100% reliable in order to avoid accident and to have a stable control of the kite under all conditions. ETH Zurich is bee developed a deep algorithm which will allow to control the kite under the most extreme situations. MIT is working in a analysis of the danger situation that an AWE system can provoce in order to come up with procedures safer and more reliables.

Alternatives to AWE technology

It has been proved that the solar energy is cheapest and widely used. In addition AWE require a lot of technical challenges. Here came up the question if it is worth to invest money in this technology. Yes it is, as this is a field with a big potential and it must be investigated. When the appropriate technology is ready, the systems will be profitable and will compensate the initial investment, while being a perfect complement to other green sources. This project has a lot of sense in the actual situation of experimenting climate change and looking to green alternatives. This technology is a great opportunity to reduce the harmful gases in the atmosphere.

3 Two-line kite model

The work in which this thesis is based on the developed of a mathematical and dynamic model of a 2-line kite linked to the ground with two tether of variable lengths. The kite is considered as a rigid body and the mathematical model is created in such a way that it can be combined with other tools like periodic orbit and non linear optimum control solvers. It analyzes the kite dynamic in cross wind condition too, driven passively or actively. Thanks to the paper "Modeling and dynamics of a two-line kite" [1], the two-line kite model can be understand. The model considered is a rigid power kite, with mass m , surface S , wingspan b and chord c . The center of mass is G , and it is the origin of S_B , the body linked reference frame. The axes of S_B coincide with the principal axis of inertia, obtaining the moment of inertia tensor in the form of a diagonal matrix. The moment of inertia tensor is the following:

$$I_G = mL_0^2 \begin{pmatrix} i_x & 0 & 0 \\ 0 & i_y & 0 \\ 0 & 0 & i_z \end{pmatrix} \quad (1)$$

The tethers are considered as a rigid bar and they are attached to the ground, in the point O_E and to the kite in the point A_+ and A_- . The rigid bar assumption can only be applied when the tension is large compared with aerodynamic and gravitational forces of the tether, and it can not be applied during phases of take off and landing. The position of the attach points in the kite are describe as following:

$$\overline{GA_{\pm}} = L_0(x_A i_B \pm y_A j_B + z_A k_B) \quad (2)$$

To describe the kite behaviour, the first step is to known the degrees of freedoms (DOF) of the system: it has 4 DOF. As the kite flight in a 3D environmental, to describe the system, 6 coordinates are required. These 6 coordinates describe the movement of S_B with respect to S_E . The two tether add two constrain to the system, as they are considered straight bar, reducing the DOF to 4. In particular, these constrains results from the attachment of A_+ and A_- to a point in the ground, O_E , providing the tether lengths: $L_{A_+}(t)$ and $L_{A_-}(t)$. The simplification of the tether is done in order to reduce the complexity in the simulator, notice that it can only be applied for short tethers. Knowing that there are 4 DOF, the equation of motion, using analytically mechanic (Lagrange formulation) can be implemented which yield to a set of ordinary differential equation. As the system has 4 degrees of freedom, 4 variables must be chosen in such a way that the system description is as simple as possible, as it will affect the complexity of the set of equations. The values used are rotation around k_E (ϕ), rotation around j_1 (γ), the rotation of S_1 with respect to S_2 (η) and the rotation of S_2 with respect to S_B (θ). These values came from the change of reference systems. The four variables chosen to describe the system result in the state of vector, defined as following:

$$x_s(\tau) = [\phi \ \gamma \ \eta \ \theta]^T \quad (3)$$

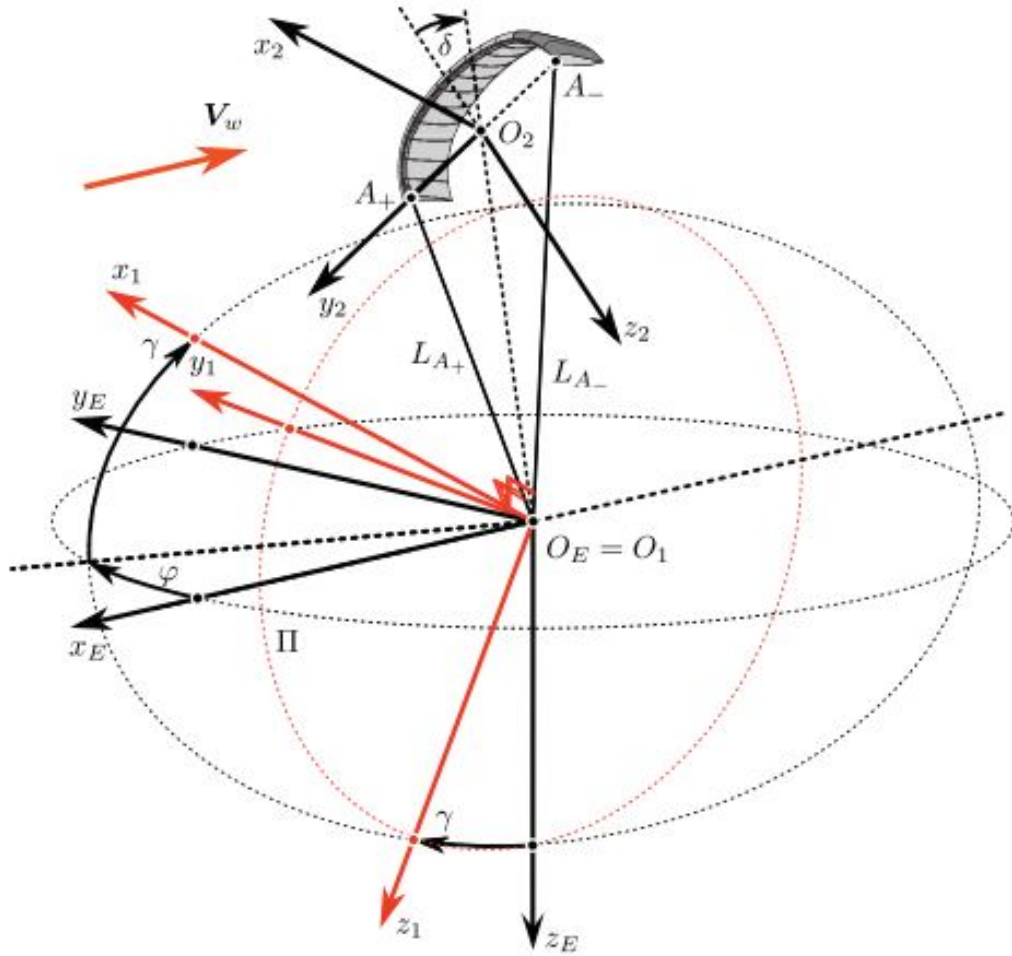


Figure 12: Kite coordinate systems [1]

In order to explain properly the reference systems, it is important to define the π plane, as it is going to be very helpful in the description of the reference systems. The π plane is the one containing, O_E , A_+ and A_- . Notice that A_+ , O_2 and A_- form always a straight line and this imply that O_2 is in the π plane too, as far as O_2 is defined as the midpoint between A_+ and A_- . In addition, an "imaginary" triangle is established, formed by O_E , A_+ and A_- . Since the length of the tethers are $\overline{O_E A_+}$ and $\overline{O_E A_-}$ and they are $L_{A_+}(t)$ and $L_{A_-}(t)$ and the distance $2L_0 y_A$ is constant. S_E is the system of reference attached to the earth, with origin in O_E . S_1 has the same origin as S_E ($O_E \equiv O_1$) but axes $O_1 y_1$ and $O_1 z_1$ spanning the π plane. The change from S_E to S_1 is made by a rotation of φ around k_E axis, and a rotation of γ around j_1 . Its rotation matrix is defined in equation 4:

$$R_{1E} = \begin{pmatrix} c\gamma c\varphi & c\gamma s & -s\gamma \\ -s\varphi & c\varphi & 0 \\ s\gamma c & s\gamma s & c\gamma \end{pmatrix} \quad (4)$$

The reference frame S_2 moves linked to the triangle explained before. Its origin, O_2 , is in the midpoint of $\overline{A_+ A_-}$. $O_2 y_2$ goes from A_- to A_+ and $O_2 z_2$ is constrained inside the π plane. Thus, $O_2 x_2$ is normal to π plane. The change from S_1 to S_2 is determined by the angle η formed between y_1 and y_2 . The rotation matrix R_{21} is shown in equation 5

$$R_{21} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c\eta & s\eta \\ 0 & -s\eta & c\eta \end{pmatrix} \quad (5)$$

The body linked system S_B is used to fully determine the kite behaviour. Its origin is the kite center of gravity "G" and x_B and z_B axis lies in the π plane, like x_2 and z_2 . The position of "G", with respect to the origin is: $\bar{r}_G = -x_A\bar{i}_2 + -z_A\bar{k}_2$. The rotation of S_2 with respect to S_B is made by an angle θ around j_2 . Its rotation matrix is shown in equation 6 and the graphical reference frame change from S_2 to S_B is shown in figure 13.

$$R_{B2} = \begin{pmatrix} c\theta & 0 & -s\theta \\ 0 & 1 & 0 \\ s\theta & 0 & c\theta \end{pmatrix} \quad (6)$$

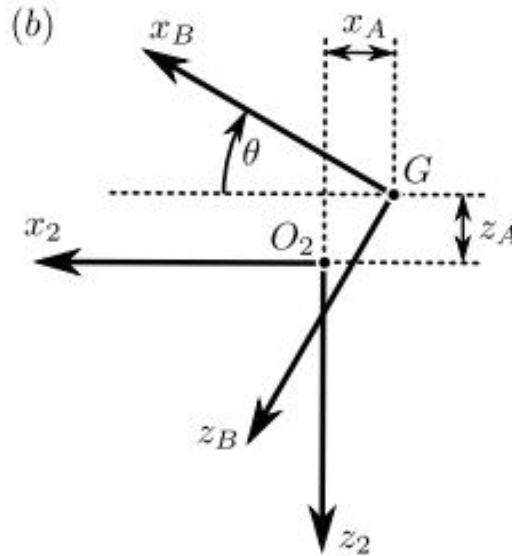


Figure 13: Change from S_2 to S_B [1]

Once the kite is defined, the control vector can be defined too. Although the kite is controlled by two tethers, and thus the natural control vector would be L_{A+} and L_{A-} , for simplicity in the calculations the control vector used is composed of l and δ , two dimensionless variables:

$$x_c = [l \ \delta] \quad (7)$$

These two values are used because they appear in all the equations, and use them is more simple than use the lengths. They are directly related to L_{A+} and L_{A-} , and the triangle characteristic. l is the distance between O_E and O_2 , in dimensionless form: $l = \frac{O_E O_2}{L_0}$, obtained from formula (8). δ is the angle between $O_2 Z_2$ and the median of the triangle

with vertices O_E , A_+ and A_- . Applying the law of cosines a formula for δ is obtained, see (9). To completely understand the equations, it is important to know that l_{A+} and l_{A-} are dimensionless: $l_{A+} = \frac{L_{A+}}{L_0}$, $l_{A-} = \frac{L_{A-}}{L_0}$ and the distance $2L_0y_A$ is constant. The value of y_A is obtained from this constrain.

$$l(\tau) = \sqrt{\frac{1}{2}(l_{A+}^2 + l_{A-}^2 - 2y_A^2)} \quad (8)$$

$$\delta(\tau) = \arcsin \left[\frac{l_{A+}^2 - l_{A-}^2}{4ly_A} \right] \quad (9)$$

3.1 Lagrangian function and forces

The matlab program use Lagrangian formulation to solve the equations. This kind of formulation has advantages with respect to the typical classical mechanic formulation, as it allows to work with more complex equations. Previously, the formulation of all relevant kinematic quantities must be explained for a good understanding of the Lagrangian formulation. The position of the center of mass is:

$$R_G = \overline{O_E G} = \overline{O_E O_2} + \overline{O_2 G} \quad (10)$$

and the velocity of the center of gravity is:

$$v_G = \frac{dR_G}{dt} = \sqrt{gL_0}v_G \quad (11)$$

to finally define the kinematic of the kite, its angular velocity is needed:

$$\Omega_{BE} = \Omega_{B2} + \Omega_{21} + \Omega_{1E} = \sqrt{\frac{g}{L_0}} (\dot{\varphi}k_E + \dot{\gamma}j_1 + \dot{\eta}i_2 + \dot{\theta}j_B) \quad (12)$$

For easier calculations the angular velocity is expressed in S_B reference frame:

$$\Omega_{BE} = P i_B + Q j_B + R k_B \equiv \sqrt{\frac{g}{L_0}} \omega \quad (13)$$

The Lagrangian equation is a second order partial differential equation and its solutions are the function which makes a function stationary. They are very useful to solve optimization problem in which a function is maximize or minimized. In the case of the kite, it is the most efficient way to describe the kite model. The dimensionless Lagrangian of the kite is:

$$\mathcal{L} \equiv e_k - u_P \quad (14)$$

As it can be observed it is related with the kinematic and potential energy. The definition of both is the following:

$$e_K = \left(v_G^2 + \frac{1}{2}\omega^T + l_G + \omega \right) / 2 \quad (15)$$

$$u_P = (-r_G k_E) \quad (16)$$

Aerodynamic forces and moment must be incorporated to the Lagrangian equation, through the generalized forces, as they are non conservatives:

$$Q_i = f_A \frac{\partial v_G}{\partial \dot{x}_{Si}} + m_A \frac{\partial \omega}{\partial \dot{x}_{Si}} \quad (17)$$

By definition, aerodynamic and torque forces are defined in the earth reference frame, as they are directly related to the wind velocity, and the wind velocity is defined there too. But in the Lagrangian equation, all the forces must be expressed in Body reference frame. After some change of variables, aerodynamic force and torque are defined as following:

$$f_A = \nu v_A^2 [(C_{x0} + C_{x\alpha}\alpha)i_B + C_{y\beta}\beta j_B + (C_{z0} + C_{z\alpha}\alpha)] \quad (18)$$

$$m_A = \nu V_A^2 [\epsilon_b(C_{l\beta}\beta + C_{lp}p)i_B + \epsilon_c(C_{m0} + C_{m\alpha}\alpha + C_{mq}q)j_B + \epsilon_b(C_{n\beta}\beta + C_{nr}r)k_B] \quad (19)$$

These equations are extremely related with the angle of attack (α) and the slide-slip angle (β), defined in equation 20 and 21:

$$\alpha = \arctan\left(\frac{v_A \cdot k_B}{V_A \cdot i_B}\right) \quad (20)$$

$$\beta = \arcsin\left(\frac{v_A \cdot j_B}{|V_A|}\right) \quad (21)$$

This aerodynamic model is only valid for values of $\alpha < \alpha_s$ (stall angle of attach) and slide-slip angles smaller than the maximum one $|\beta| < \beta_{max}$. This is a basic aerodynamic model, but in future researches a more complex model can be developed, and they can be easily introduce in the calculation substituting equation 18 and 19. All the parameters using in the simulation are shown in the table 1, and came as the result of the investigation "Flight Dynamics and Stability of Kites in Steady and Unsteady Wind Conditions" [12].

Symbol	Value	Symbol	Value	Symbol	Value
ϵ_c	0.0075	ϵ_b	0.029	ν_W	0.25
x_A	0.0037	y_A	0.0037	z_A	0.01
i_x	$1.32 \cdot 10^{-4}$	i_y	$2.92 \cdot 10^{-5}$	i_z	$1.12 \cdot 10^{-4}$
μ	440.4	β_{max}	15°	α_s	25°
C_{x0}	-0.065	$C_{x\alpha}$	0.176	$C_{l\beta}$	-0.1
C_{lp}	-0.15	$C_{y\beta}$	-1.57	C_{m0}	0.1332
$C_{m\alpha}$	-0.7633	m_q	-0.165	C_{z0}	0.116
$C_{z\alpha}$	-2.97	$C_{\eta\beta}$	-0.027	$C_{\eta r}$	-0.002

Table 1: Dimensionless parameters used in the simulation [12]

3.2 Equation of motion

The equation of motion can be defined thanks to the Lagrangian equation. The Lagrangian equation can be written as:

$$\frac{d\mathbf{u}}{d\tau} = \mathbf{f}(\mathbf{u}, \tau; \mathbf{p}) \quad (22)$$

$\frac{d\mathbf{u}}{d\tau}$ depends on the extended state vector \mathbf{u} ; τ , (due to the fact that Lagrangian equation depends on the control law, and the control law depends on τ); and \mathbf{p} , which contain all dimensionless parameters, including the aerodynamic ones (see table 1). The equation of motion formula leads to 23:

$$\frac{d}{d\tau} \left(\frac{\partial \mathcal{L}}{\partial \dot{x}_{si}} \right) - \frac{\partial \mathcal{L}}{\partial x_{si}} = Q_i \quad (23)$$

with $i=1,\dots,4$

Finally, combining equation 11, 20 and 21 it is created a system of first ordinary differential equation, where $\frac{d\mathbf{u}}{d\tau}$ is defined. \mathbf{u} is the extended state vector $\mathbf{u} = [x_s \dot{x}_s]$, where $x_s = [\phi \ \gamma \ \eta \ \theta]$. It must be notice that y_A does not appear in it, but this value is really needed to make the change from l_{A+} and l_{A-} to δ and l . The main advantage of working with non dimensional values is that this model is valid for any rigid body towed to the ground by two tether, so it can be easily applicable to different kites (f.e. acrobatic, power or aircraft-like kites). For sure, the result is not the same with one kite or another, this difference is due to different shape of the body, in particular, the shape has influence in the aerodynamic coefficients and normalized moments of inertia. It is important to mention that the stability of the system depends on Cl_β , in particular, it has influence in the lateral dynamic of the system. Cl_β value can be tuned in the kite design, modifying dihedral and sweep angle, and a proper system can be achieved, but this analysis is out of the scope of this thesis.

Dynamic of the kite

The kite dynamic depends on the control law, for different control laws, the kite experience different behavior. It is important to mention two kind of control laws, which are used in the development of the Simulink program and has effect on the stability: no control and variation of tether length periodically, varying δ . From the first case, it is known that a kite can fly stable in crosswind condition with no control, which will be the case in some part of the Simulink simulation. The stability of the equilibrium position depends on Cl_β , which is a value related to the kite design. In the second case, the tether length change periodically, varying δ as function of the time, and the following control law is used: $\delta(\tau) = \delta_0 \sin(\omega_\delta \tau)$. From this investigation it was observed that in the case of kite with variable tether lengths, Cl_β values have bigger effect on the stability than in case with constant tether length, but stable trajectories can be achieved too.

3.3 Limits of the model

By numerical integration, $u(\tau)$ is obtained. Although this result is mathematically correct, it must satisfy some physical limitations:

- The tethers must be under traction at every time. To check it, the tension force at points A_\pm is: $T_\pm(\tau) = mgt_\pm u_\pm > 0$. To have a physical valid trajectory, $T_\pm(t) > 0$ at every t .

- Aerodynamic constrains: $\alpha < \alpha_s$ and $|\beta| < \beta_{max}$

To check the veracity of this system, in LAKSA program, the physical model is been solve using two methods: Classical Newton-Euler, applied to a rigid body with the tether constrains and Lagrangian formulation. Both provide similar result, up to little errors due to numerical integration, but Lagrangian formulation is chosen as the one for use in the development of the simulator for different reasons: it works better with complex calculation, f.e. working with periodic orbits, and it requires less computational time, which is ideal for achieve real time simulation.

4 Simulink program

This thesis has developed a Simulink model for the simulation of a two-line kite with control inputs in real time. The version of Matlab used in this thesis is MATLAB R2018a. During the thesis an implementation methodology has been followed, where each phase adds a new feature with respect to the previous iteration (user input, real time simulation, kite display...), starting with the recreation of LAKSA code in Simulink. In addition this will allow the program to have a modular structure, so at any time, some parts of the simulator can be improved, like the aerodynamic model, the control law..., and easily implemented. The final goal is to obtain a Simulink program, based on LAKSA one, with more features. In particular the goal is to control, simulate and plot the kite at real time. The kite parameters using in this thesis are defined in table 2.

Chord (c)	1.5 m	Span (b)	5.8 m
Mass (m)	4 kg	Surface (A)	14.4 m ²
I _x	21.1 kg/m ²	I _y	4.66 kg/m ²
I _z	18 kg/m ²	I _{xz}	0 kg/m ²
α _s	25°	β _{max}	15°
C _{x0}	-0.065	C _{xα}	0.176
C _{lβ}	-0.1	C _{lp}	-0.15
C _{yβ}	-1.57	C _{m0}	0.1332
C _{mα}	-0.7633	m _q	-0.165
C _{z0}	0.116	C _{zα}	-2.97
C _{ηβ}	-0.027	C _{ηr}	-0.002
Earch acceleration (g)	9.81 m/s ²	ρ	1.225 kg/m ³
Wind velocity	7 m/s	Tether length (L ₀)	80m

Table 2: Kite parameters [12]

4.1 Recreate LAKSA code in Simulink

To check the correct implementation of the Matlab code into Simulink, this thesis has developed a Simulink model which recreate LAKSA code, with the same analytical control law, in order to compare the result of both simulations. The Simulink set up can be seen in figure 14. This program has different components, being the basic one "ODE_Control_Analytical", as the Lagrangian equations, and thus the kite behavior, are defined in it and it determines the state vector, \mathbf{u} .

$$\mathbf{u} = [\mathbf{x}_s \ \dot{\mathbf{x}}_s] \quad (24)$$

$$with \ \mathbf{x}_s(\tau) = [\phi \ \gamma \ \eta \ \theta] \quad (25)$$

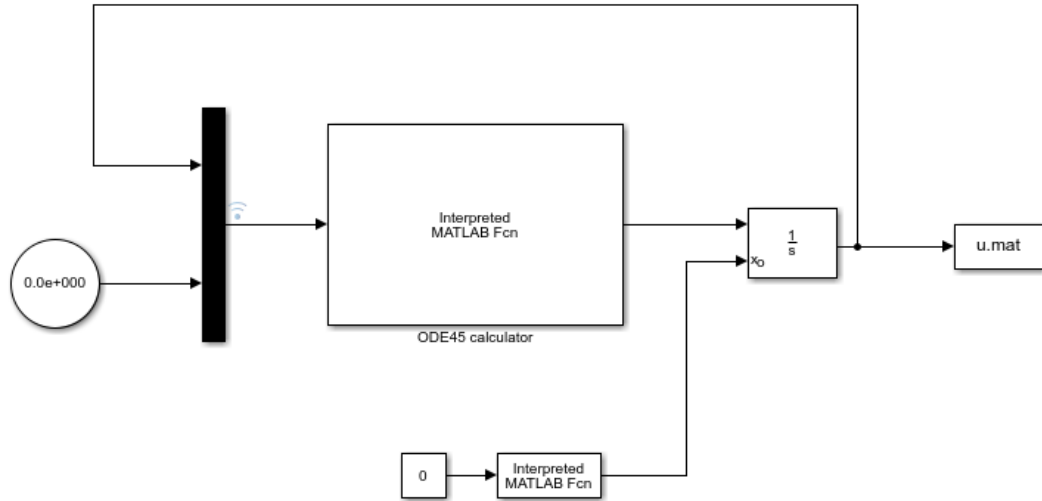


Figure 14: Simulink model [9]

At the time to face the implementation of LAKSA program in Simulink, different problems came up. One of the most important problem to deal with was the fact that LAKSA code use struct variables (like PD or PND, where all kite characteristic are defined), but Simulink does not deal with it. The first attempt was to change all struct variables into vectors, but this solution was rejected due to complexity. After some investigation, the possibility of use the block "Interpreted Matlab Function" appear and, from the very beginning, it fixes perfectly with the requirements: it can deal with struct variables, as far as they are created at the beginning of the function, and it allows to use LAKSA functions without big modifications. The block "Interpreted Matlab Function" basically execute the function assigned to it. Another important decision was to chose how to integrate. Taking advantage of LAKSA funtion, integration can be carried out inside the function, with "ode45" command, being the same integration command as LAKSA. The other solution is to use "Integrator" block, of Simulink. A model with each solution was tested, and although both works, the solution which integrate inside the function was rejected, because it add complexity to the model and as it is not the natural way of doing an integration in Simulink, in the future can be a source of problems. That is why "Integrator" matlab box was chosen. It has a plenty of configuration options, so it can recreate LAKSA integrator. One of these options is the possibility to chose a solver: ode45 (Dormand-Prince) was chosen, as it is the most similar to the one used in LAKSA program, "ode45".

The essential of this program is the function "ODE_Calculator". This function solve the Lagrangian equations, taking advantage of LAKSA functions Fun_ODE_Lag_KA, and output the derivative of the state vector, which is integrated to obtain the state vector. The control vector is defined inside the function Fun_Control_KA, analytically in this case, and it has direct influence in the Lagrangian equations. The structure of "ODE_Calculator" is the following:

- Split the input in known variable: u , t . This is a direct consequence of the fact that Interpreted Matlab functions has only one input port.
- Define constant values: PD and PND

- Obtaining the derivative of the state vector, $\frac{d\mathbf{u}}{d\tau}$, which is the output, using LAKSA functions.

In this model, the state vector depends on the time, in order to recreate the control vector of Matlab code: $x_c = [l \ \delta]$, being l constant and $\delta = \delta_0 \sin(\omega\tau)$. In section 4.2, a control command is implemented, allowing the user to input δ . The main advantage of using Interpreted matlab function is that Fun_ODE_Lag_KA can be used without modify the original LAKSA function. In addition, struct variable can be used, like PD and PND, as far as they are defined at the beginning of the function. The block "Integrator" requires an initial value to simulate, and like in LAKSA program, it is the equilibrium state of the kite. It can be obtained from "Equilibrium_KA.m", but as "Interpreted Matlab function" need an input and an output, this function must read a constant 0 as input, although this value is no longer used. The output is the state vector \mathbf{u} at equilibrium position, \mathbf{u}_0 . The other blocks used in the program are clock and to_file. Clock block generates time signal and To_file save values at every time step, in this case the state vector is saved which is very useful to compare the result with the one obtained from Matlab and check the reliability of the results.

To have good result, a proper Simulink set up is important. Simulink set up aims to reproduce the Matlab set up in the ode45 solver, which set the tolerances and time step. The configuration of Simulink simulation parameters is done in the box "Configuration Parameters" (See 15). Simulink solver is ode45, and the relative and absolute tolerances are 10^{-3} and 10^{-6} , respectively. Regarding the step size, the original Matlab code simulate with fixed step size, but with Simulink this is no longer possible. Instead, the maximum and initial step size is set equal to the Matlab step size and the min step size is set to auto. In this particular simulation, at some points, Simulink require very little time step; for that reason, the min step size is auto (which is the same as no minimum step size) otherwise, a minimum time step will lead to errors.

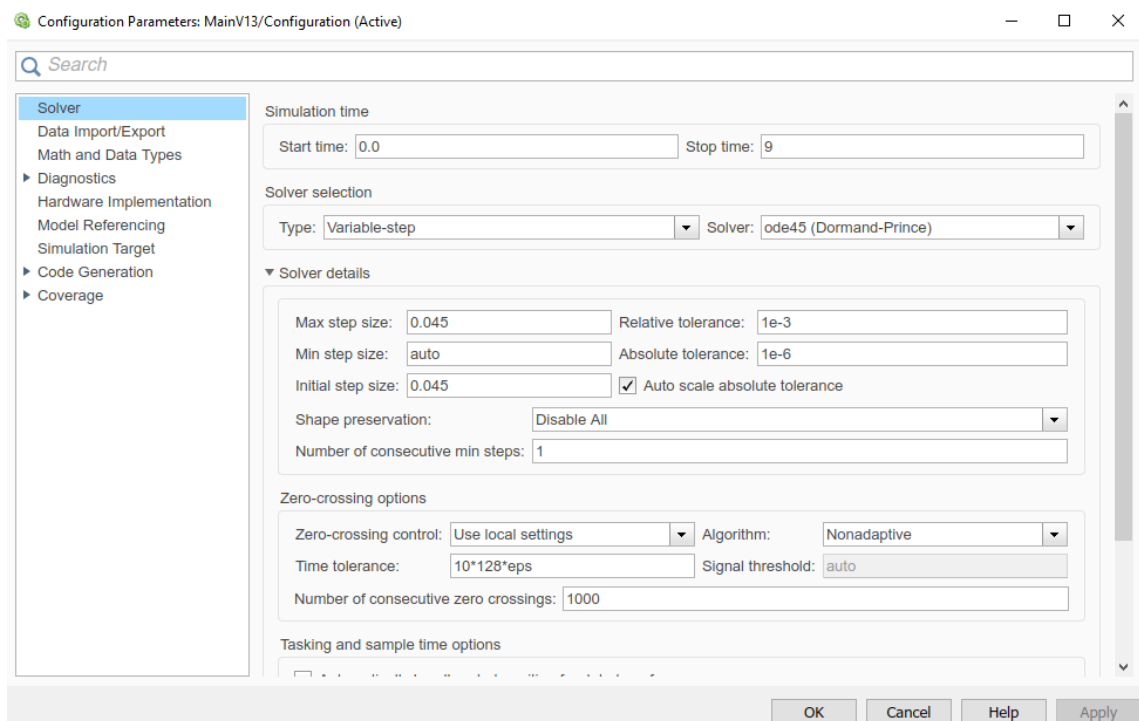


Figure 15: Solver options [9]

4.2 User input

This thesis developed control inputs in real time for the kite control. The control of the kite is made modifying the tether length L_{A+} and L_{A-} , but in order to simplify the calculations, these lengths are expressed with other two variables: δ and l , defining the control vectors as follow:

$$x_c = [l \ \delta] \quad (26)$$

and the derivatives of the control vector, which are strongly used in the calculation of the Lagrangian equation:

$$\dot{x}_c = [\dot{l} \ \dot{\delta}] \quad \ddot{x}_c = [\ddot{l} \ \ddot{\delta}] \quad (27)$$

In this thesis, l is kept constant, so the control of the kite depends only on δ . Although the thesis goal is to control δ , in the development process three different models were used, where the control is made via δ , $\dot{\delta}$ or $\ddot{\delta}$ and the other two are obtained by derivation or integration. The different models can be observed in figures 16, 17 and 18. The user input the values using a knob, connected to a constant block, which generate a signal equal to the knob value. Notice that although the box is called constant, its output value is not. In version 2 and 3 there is a box between the input and the output, see 17 and 18. The goal of this box is to set the values of $\dot{\delta}$ and $\ddot{\delta}$ to 0, when δ reach its maximum or minimum value. The value of δ is limit in these versions thanks to the option of the integrator block, but it does not set to zero the other values, which should be zero when δ is constant. In all these models Real-Time Synchronization was implemented. It allows to control the simulation time, as the input modules do not require too much computational time, the simulation run too fast, and it is not desired. Although digital clock is used at the beginning of the development (it can be observed in the figures), when the control model is implemented in the simulation it is changed to normal clock. During the development process it was consider whether use digital or normal clock, so different models with each clock were tested, and the normal one was chosen, as it was the only which gives no errors performing the integration.

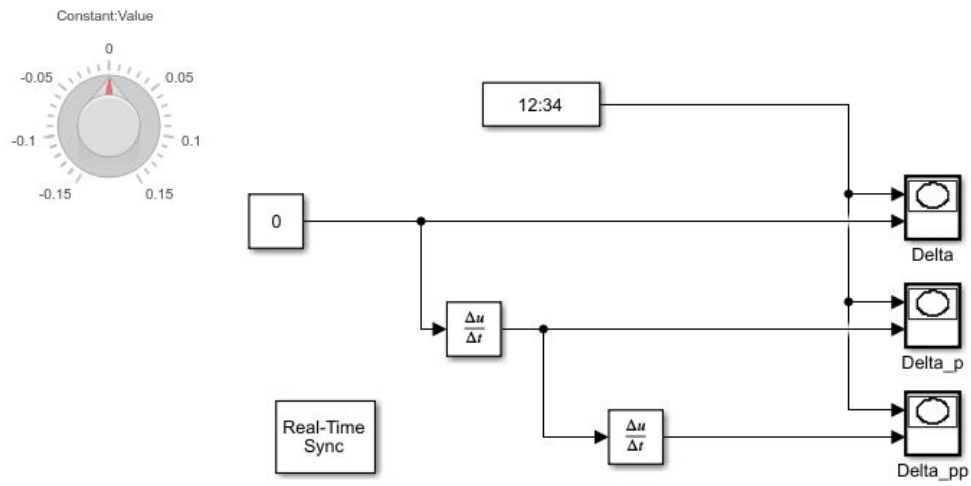


Figure 16: Joystick V1.1 - δ [9]

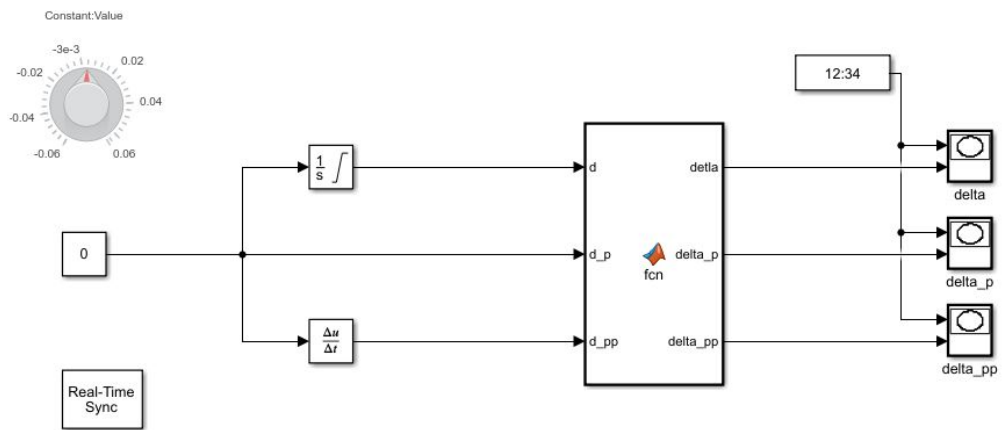


Figure 17: Joystick V1.2 - $\dot{\delta}$ [9]

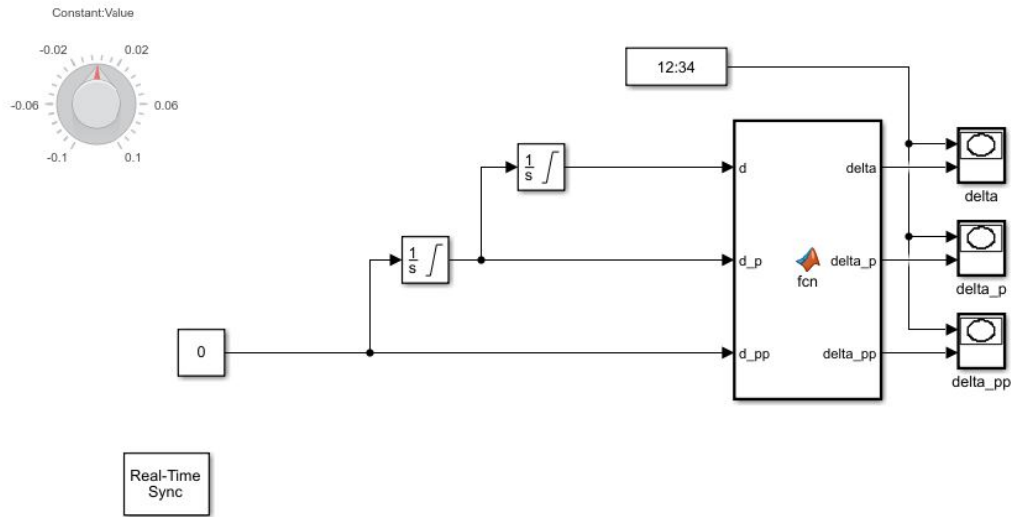


Figure 18: Joystick V1.3 - δ [9]

In order to influence in the formulation of the Lagrangian equation, with the control law input by the user, the Simulink program presented in section 4.1 is modified. The resultant program is shown in figure 19. Some modifications must be done: "ODE_Calculator" must read δ and its derivatives values, and use them to create x_c , \dot{x}_c and \ddot{x}_c . Notice that \dot{l} and \ddot{l} are 0, as l is constant. δ values are saving in a file called delta.mat, so this values can be used to recreate Simulink simulation in LAKSA program. Fun_ODE_Lag must be modified too, as now it has as input the control vector and it must not call Fun_ODE_Control, to create it.

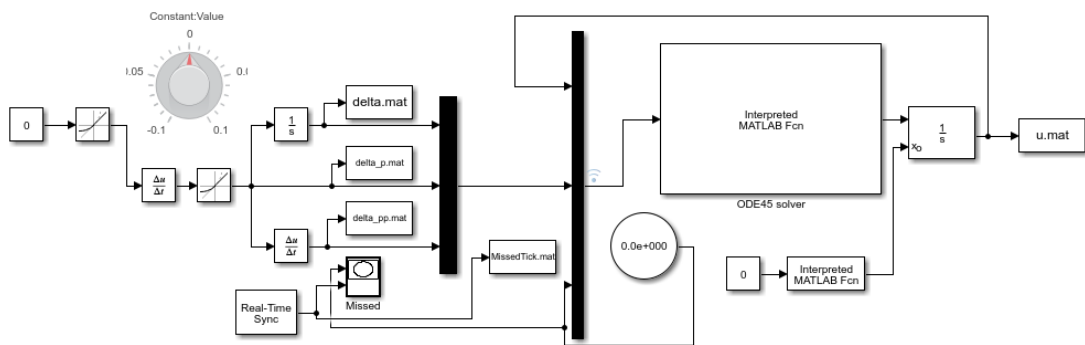


Figure 19: Simulink final model [9]

Parallel to the Simulink development, LAKSA Matlab code is modify in order to read the control values from the Simulink simulation, saved in delta.mat, and perform the simulation with them. It is useful to compare its result with the Simulink one and check the veracity of the Simulink simulation. After performing simulations with different set up for each control model, just the simulation with control in δ gives no errors. The result of δ simulation were compared with the LAKSA one and it confirms that the model produce reliable results. Although the concept works, δ controller need to be improved in order to allow the user to input δ . In particular the problem in the simulation when the user input

4 SIMULINK PROGRAM

δ or $\dot{\delta}$ is due to numerical computation, as consequence of big changes in δ and $\dot{\delta}$. In order to solve it, a filter, "Rise Limiter" is used.

δ model

δ rise must be limited, in order to avoid numerical problems in the computation of the Lagrangian equations. It can be solved using "Rise Limiter" block. In figures 20 and 21 a model without and with a filter are shown, with a plot of the value of δ and its derivatives. In the first one, it can be observed high peaks in $\dot{\delta}$ and $\ddot{\delta}$ values, which are the source of errors.

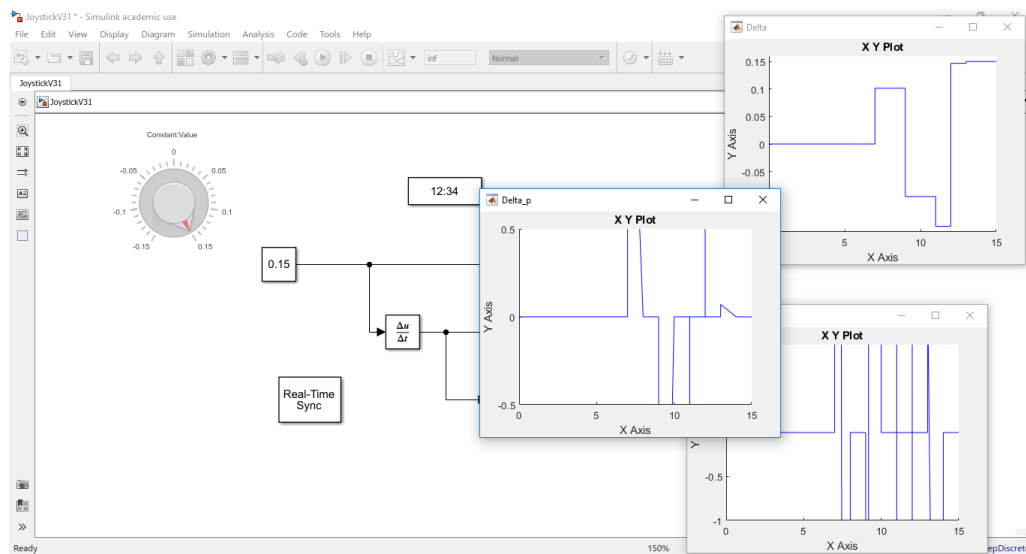


Figure 20: First model with δ as input [9]

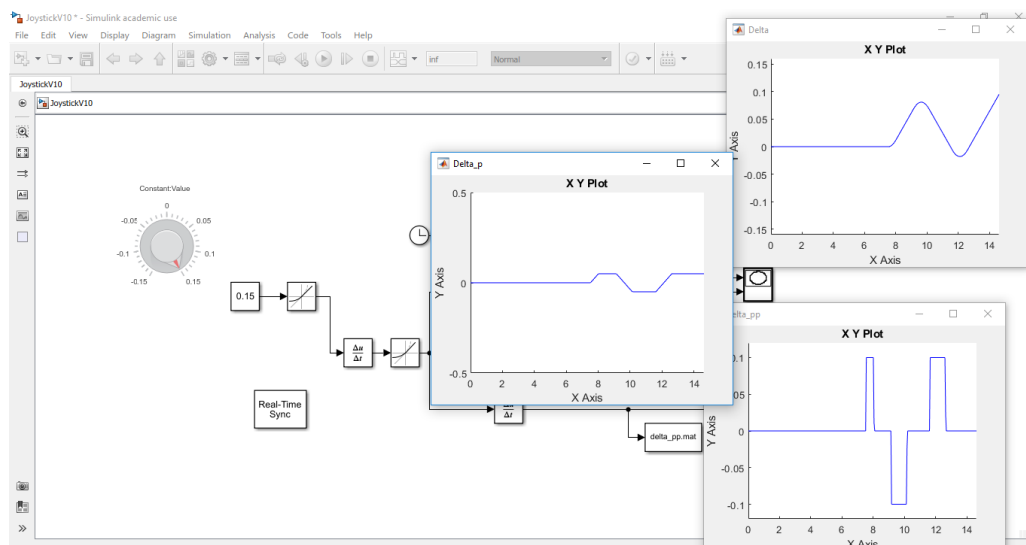


Figure 21: Final control model with δ as input [9]

In order to solve this problem, the block "Rate limiter" is used to limit the rate of

change in δ and $\dot{\delta}$, which result in limiting $\dot{\delta}$ and $\ddot{\delta}$ values. Before reaching this solution, a function was developed, which set the δ and $\dot{\delta}$ derivatives to a maximum value, making a comparison of the variation of δ and $\dot{\delta}$ with time. This work well limiting $\dot{\delta}$, but for limiting $\ddot{\delta}$ it turned to be so complex and it does not work properly, so this solution was neglected. "Rate Limiter" block is the solution chosen. It limits the increase of the signal across its input, and output the maximum allowable value. The block allows to set a maximum value for $\dot{\delta}$ and $\ddot{\delta}$. These values has influence in the physics of the kite, f.e. they are strongly related to β value, that is why the physical limits of the model will limit these values. δ must be limited too, as the change in tether length is limited. It will be explained in detail in further sections. The final model in Simulink is shown in figure 22.

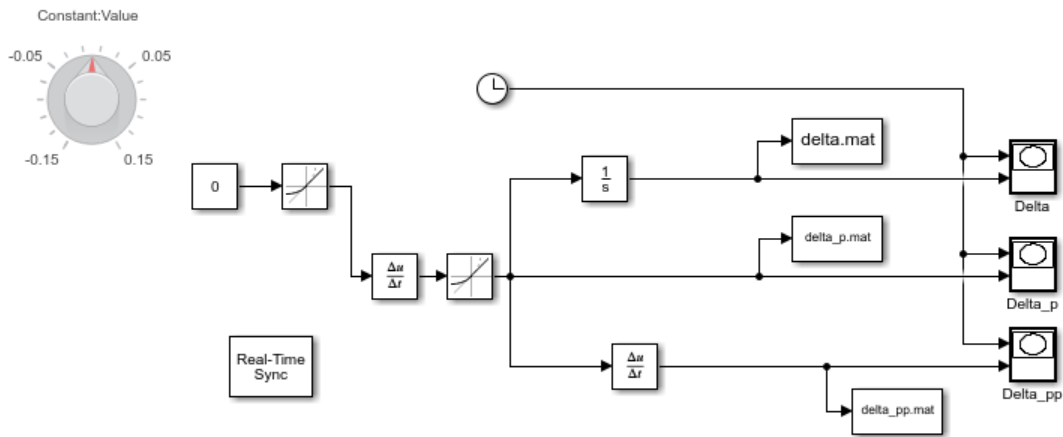


Figure 22: Controller final model [9]

4.3 Verification of Simulink calculation

To check the Simulink results, a new version of the LAKSA program has been developed. In this version, the matlab program simulate with the control vectors used in Simulink, saved in delta.mat. In order to use Simulink values, some modifications must be done in the matlab code: Fun_Control_KA is modify in order to read from delta.mat the values of δ and its derivatives, at every time in the simulation, and use them to create the control vector. As in each time step the simulation time is not the exact time as the one saved in delta.mat; the values of δ and its derivatives at every matlab time are obtained by linear interpolation, with the function "interp1". The values obtained in the interpolation, the ones that matlab uses for simulate, are saving in delta_matlab.mat, which is useful to check the veracity of the interpolation and confirm that both codes are using the same control law. An scheme of the verification process is shown in figure 23.

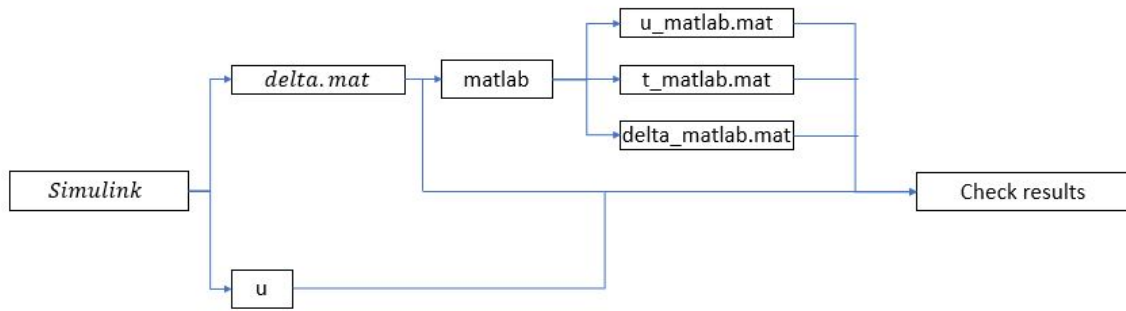


Figure 23: Verification process

As in the original code, the state vector is obtained using ode45 function. The code must be modified, in such a way that tspan is determined by the Simulink simulation time, which is saved in delta.mat, while the time step is the maximum Simulink step time. During the development process, simulations and results in Simulink were obtained in non-dimensional and dimensional form, and for this reason, two different versions were created, one in which the program read values from delta.mat in dimensional form, and another in non-dimensional. As the goal is to modify the LAKSA program as less as possible, in the dimensional case, the values are adimensionalized, at the moment they are loaded, and in the non-dimensional case no further modification are needed. The values for u , t , δ and its derivatives calculated by LAKSA are saved in these variables: `u_matlab`, `t_matlab` and `delta_matlab`, all in ND form.

For an easy analysis of the results, the state vectors from matlab and from simulink, are loaded and plotted in a different code to compare them. Control vector from both program are compared too, in order to check that they are using the same control law. Like in the previous case, it must be consider the dimensional and non-dimensional Simulink results, so when dimensional data is loaded, it must be adimensionalized, while the data obtained from matlab is always in non-dimensional form. This program always plot ND data. An example of the figures plotted can be observed in figure 24

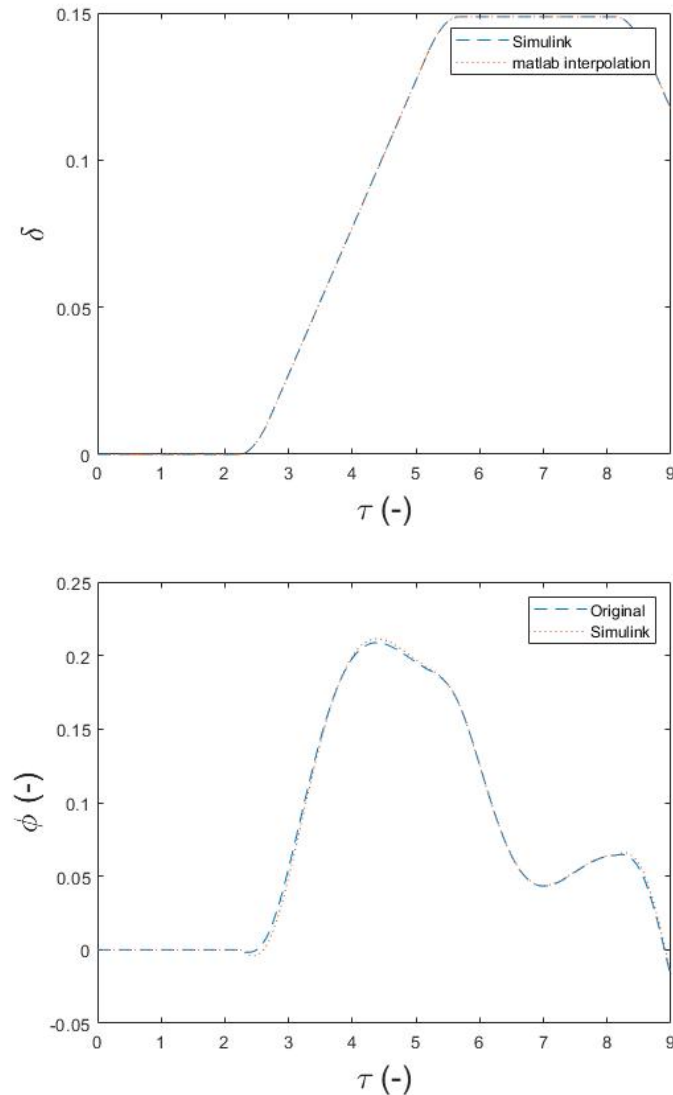


Figure 24: Example of verification process [10]

4.4 Real time simulation

When the veracity of the calculation of the Simulink model was proved and the user input was correctly implemented, a model which simulate at real time is developed. The achievement of real time simulation is based on the use of the box "Real Time Synchronization". This box allows to set a ratio between the simulation time (or sample time) and the real time. At this point one decision must be made: use a program which work with ND values in the Simulink workflow, (f.e. the signal generated by the clock is non-dimensional) and tune it to perform in dimensional form with "Real Time Synchronization" block or use always dimensional values in the workflow. The second option was chosen, due to simplicity. It is remarkable the option of "Real Time Synchronization" block called missed ticks, which measure of how good is the real time synchronization. High missed tick means that the simulation can not be performed at real time, because the computational speed is not enough, which mean that there is lag during the simulation. During the simulation

is usual the existence of parts with some missed ticks, due to a temporal lag, but as far as they go to zero, the program is simulating at real time. An example of the graph display by Simulink is shown in figure 25. From this graph it can be observed that from $t=0$ s to 4s, the program is not performing at real time, but from $t=4$ s the simulation has only temporal lags.

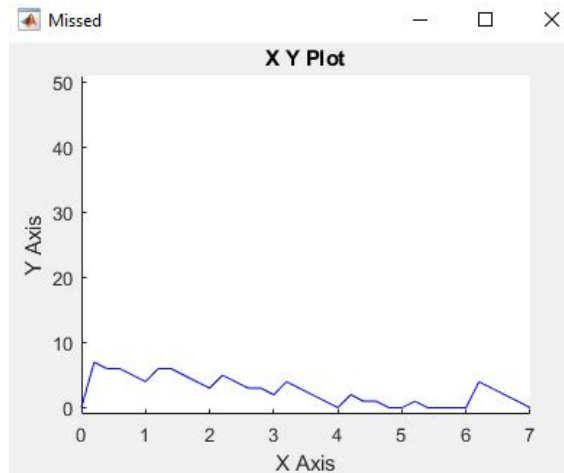


Figure 25: Mixed Ticks plot, Simulink graph (x-axis: time (s), y-axis (missed ticks (-)) [9]

To simulate at real time, a dimensional program need to be developed, which is based on the one explained in the section 4.1. The Simulink set up can be observed in figure 26 and it is the same as the explained before, with the different than the signal has dimensional values. "ODE_Calculator" is modify in order to read dimensional values for delta, its derivatives, state vector and time, at the beginning of the function, and then they are adimensionalized. The adimensionalization avoid the modification of the Lagrangian equations formulation in Fun_ODE_Lag; which is not desired, as they make the calculations in a very efficient way. After performing the calculations, Lagrangian equations return values of the state vector in ND form, which are dimensionalized and they are the output of "ODE_Control_Analytical". To check the veracity of the results, LAKSA program simulate with the control vector obtained from Simulink, and the result for the control law and the state vector are compared. The result can be observed in the figures 27 and 28. Some little errors can be observed, they are due to the difference in numerical calculation method, but they do not compromise the veracity of the results.

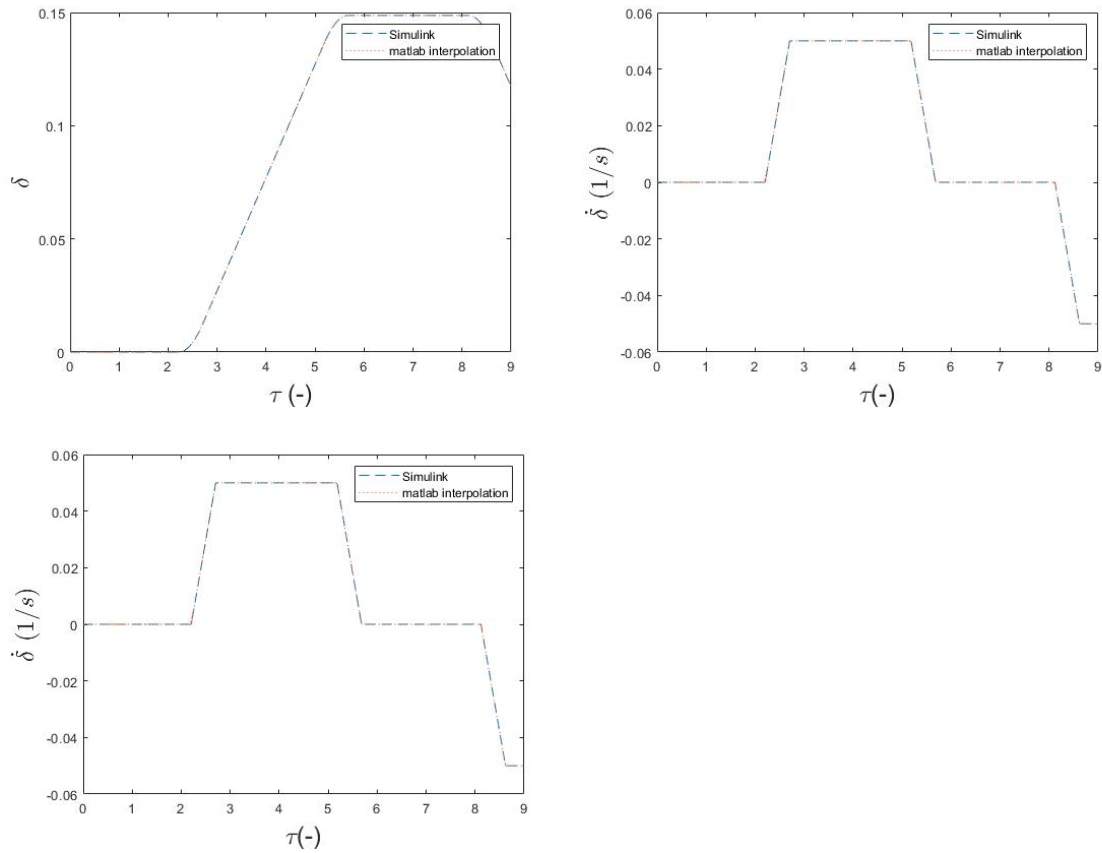


Figure 27: Comparison of δ values [10]

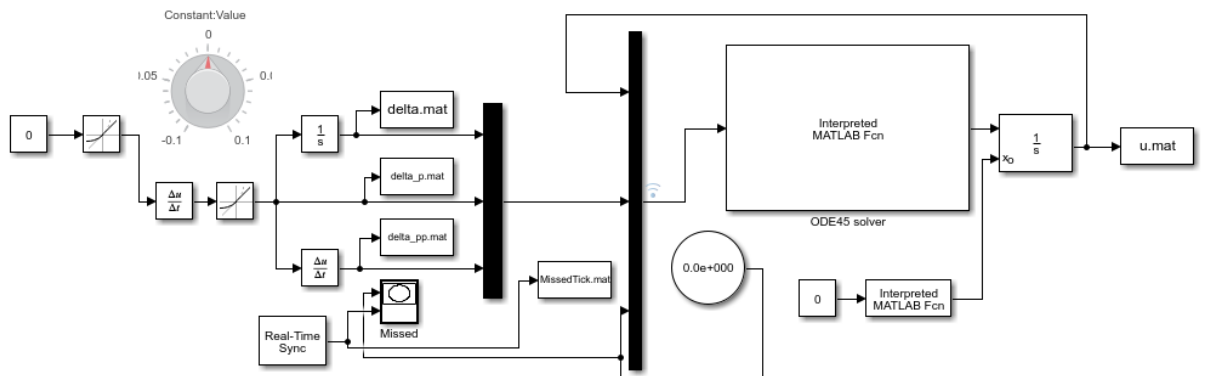


Figure 26: Simulink model, with dimensional signal [9]

4.5 Kite display

This research implement the display of the kite at real time. This is an important task because it allows the user to see the state of the kite and to control it at every time, unlike LAKSA program, which the user can only see the kite at the end of the simulation; for all this reasons, it must reproduce faithfully the kite position. In addition, it must

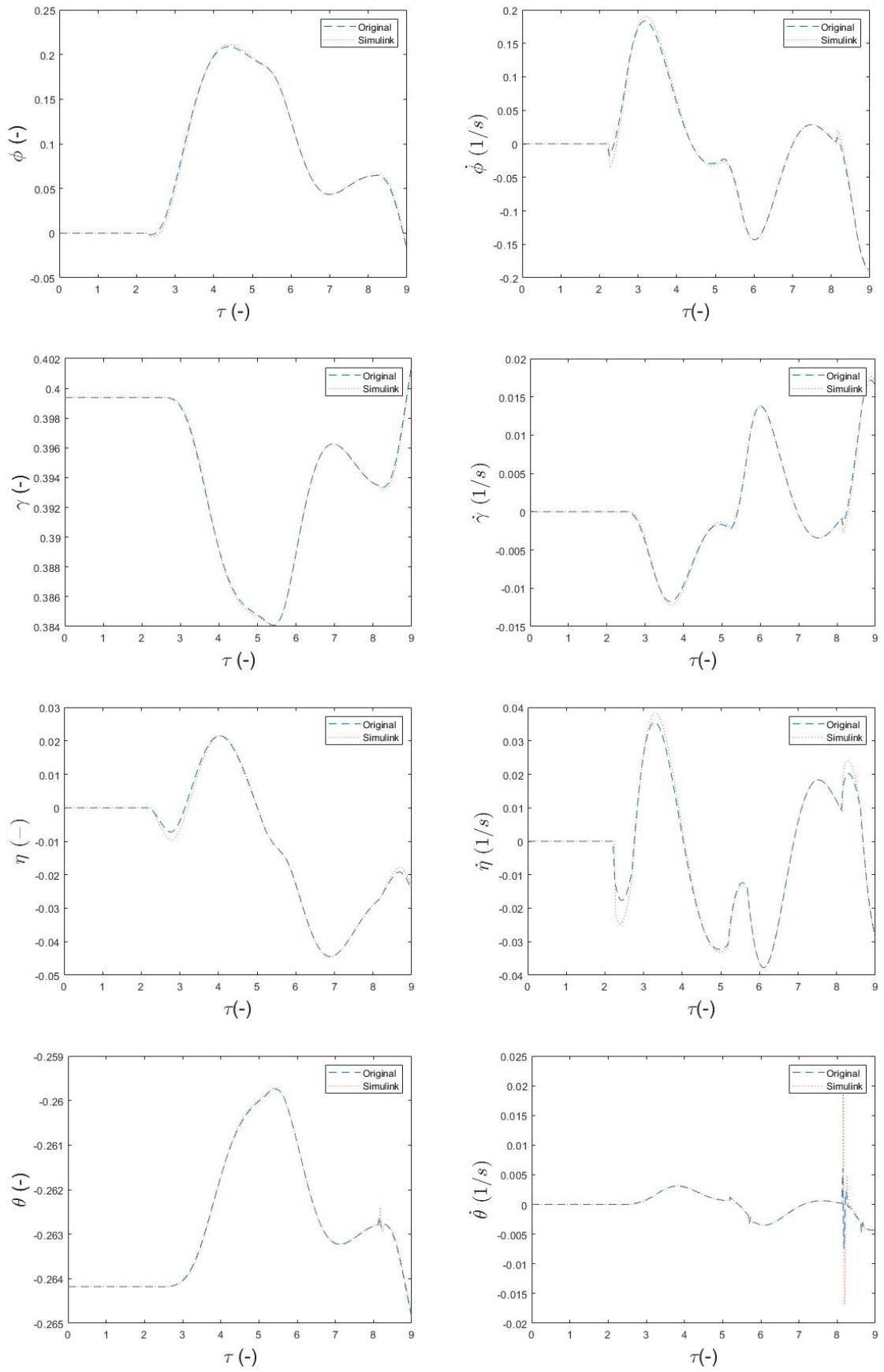


Figure 28: Comparison of state vector values [10]

be done in an efficient way, in order to save computational time and achieve real time simulation, as this is the part of the code that require most of the computer resources. The functions of LAKSA program where the kite is plotted are mostly used, although some little modifications are done. The whole process of the kite display is done in a new function "Display_KA" and in order to implement it in Simulink program, a new "Interpreted Matlab Function" block is added. Notice that this block has no input, but "Interpreted Matlab Function" force to use one, and that is why the signal end in a terminator block. The final simulink model can be observed in figure 29. Inside "Display_KA", Fun_Post_KA is used to obtain kite geometrical parameters, and with Plot_KA the kite is display. Both functions read ND values although the plot is made in dimensional form. For this reason, the input is adimensionalized at the beginning of Display_KA. Save computational time is very important in this section, as this is the part that use most of the computer resources. In order to reduce computational time, the following modifications are done:

- Inset kite plot is plot in a different figure: In LAKSA simulator, inside the figure two plots are done, one of the whole kite and other focus on the cloth. This is computational time consuming, so in order to simplify the kite display they are divided in two figures. The old and new display are shown in figures 30 and 31.
- Axis initialization: Unlike LAKSA simulator, in Simulink, the axis initialization is done only once, at $t=0$. The reference matlab code initialize the axis and set the graphs at every time step, in such a way that as the simulation time increase, it requires more and more computation per step.
- Plotting the kite at a defined time step: At some parts of the simulations, time steps are too small, and plotting at every step requires too much computational time. To solve this problem "Rate Transition" block is implemented, right before Display_KA function, which allows to execute the block which is following it at a different rate, using the input signal at that specific moment. This value must be chosen properly, a small value will increase the computational time, and thus will influence the real time synchronization and a big value will display the kite change at bigger times step, being difficult to control the kite. The final Simulink set up can be observed in figure 29. Another solution to this problem were considered, like the creation of a code which compare the last time the kite was plot and the actual time, and only execute the plot commands when the time difference is bigger than the defined one. This solutions was neglected, because the Simulink code does not behave properly.

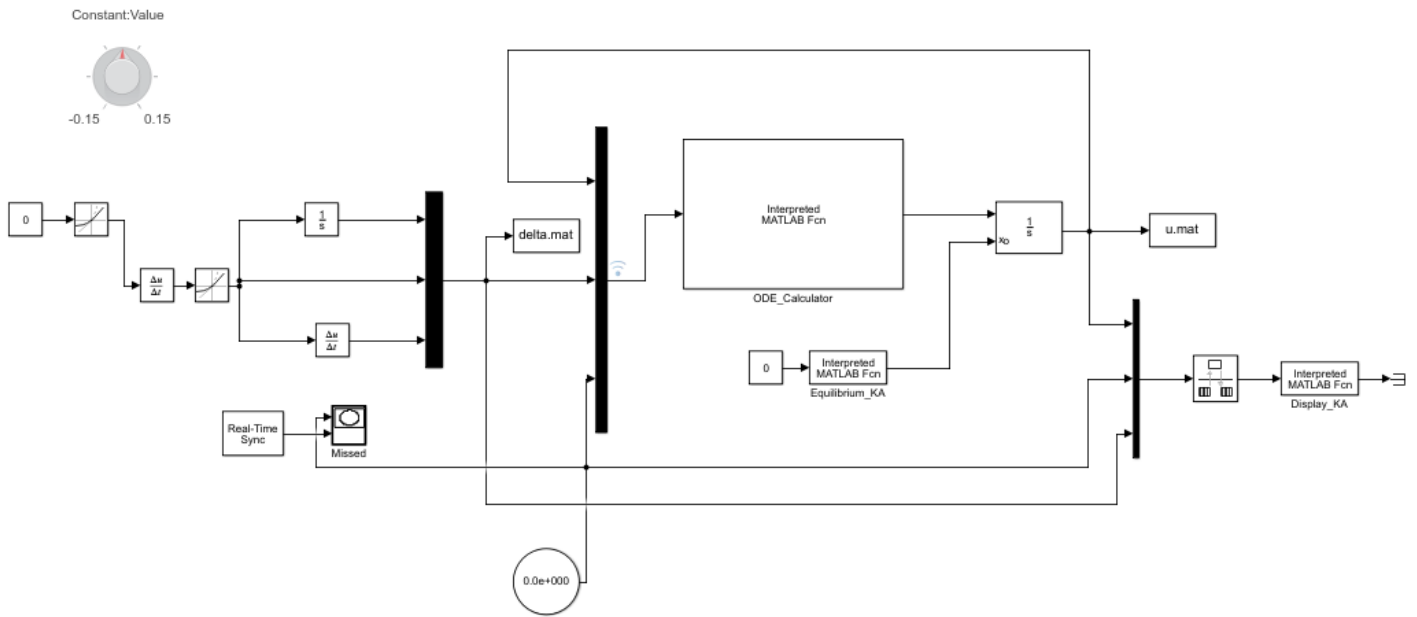


Figure 29: Simulink final model [9]

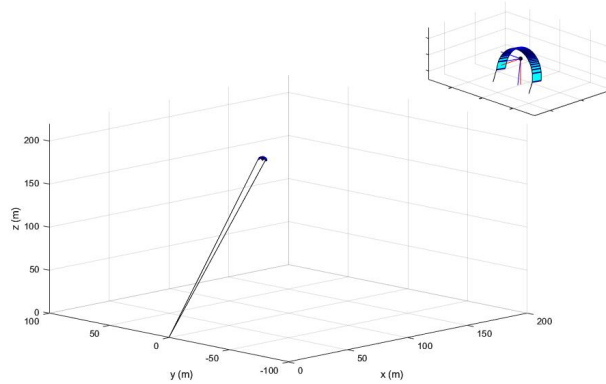


Figure 30: LAKSA Display [8]

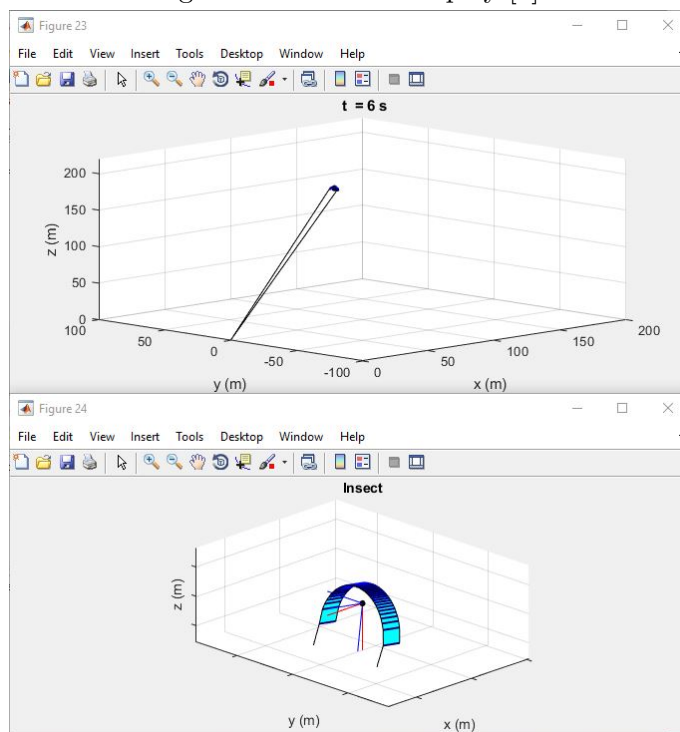


Figure 31: Simulink Display [9]

Although the essential modification for a good behaviour of the program are the ones explained before, another modification is done, in order to have a more user friendly program. The plot code is modify to place the figures in a defined space in the window, in particular in the right part. With this modification the window can be divided in 4 parts, containing all the information needed: knob, missed tick graph and kite display. Notice that the knob and the missed tick graph must be placed manually, but the position is kept every time the simulation is launched. Regarding the kite plot, every time the simulation is launched, the figures are place in a defined position, if they are moved by the user, when the simulation start again, they will move to the defined position, that is why this modification is so important. The window configuration can be observed in figure 32

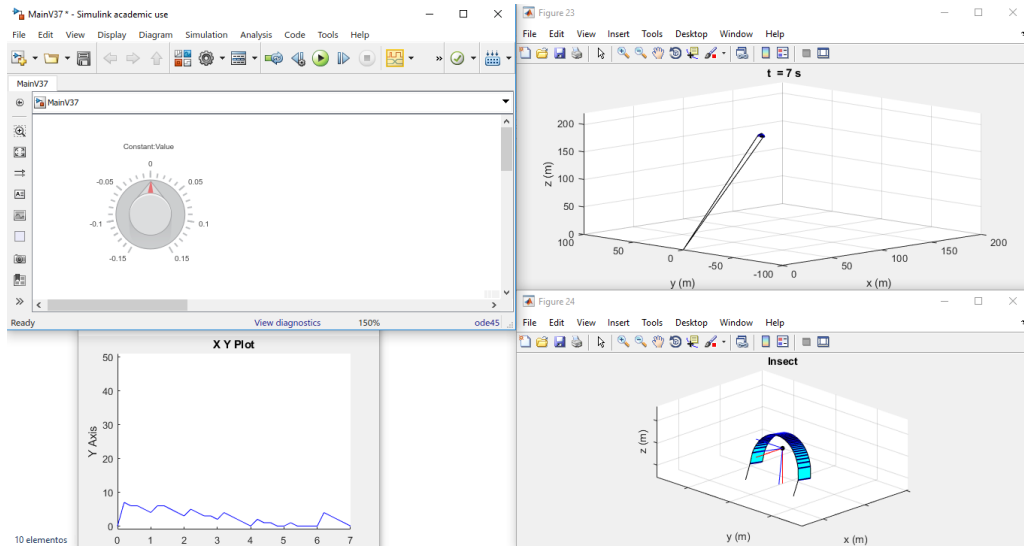


Figure 32: Simulink Window set up [9]

4.6 Physical consistency and influence of the control law

The kite model makes some assumptions which add some limitations to the model. In this section, it is going to be explained how the data obtained from the simulation is used to check that the results are not only mathematical but physical. Using state and control vector, all physical parameters of the kite can be obtained, in particular it is interesting to analyze the values of the tether forces (F_{A+} and F_{A-}), angle of attach (α) and slide-slip angle (β). It is convenient to analyze the results independently of the simulation, in order to reduce the computational time of it. Although, these data are obtained while displaying the kite, they can not be saved and used because they are not calculated at every time step, and some information would be missing. To analyze the values, a matlab code called *Kite_Physic* is created, which reads the values of the state and control vector, and with the function *Fun_Post_KA*, all physic parameters are obtained at every time step. Once the parameters are obtained, the ratio between the values and its maximum ones are plotted with respect to time in one graph. This procedure is chosen because it allows to check all the values at one look: in case that a value is greater than 1, or smaller than 0, the simulation is outside the physic limits. In case of β , negative values are allowed, in this case, its absolute value is plotted. The maximum values for β and δ are the one inherit from LAKSA program, $\alpha_s = 25^\circ$ and $\beta_{max} = 15^\circ$ [12]. For the maximum values of the forces, a value of 600N has been chosen. This value is chosen in accordance with the maximum forces obtained from the simulation, after a safety factor, in order to have the tether forces always inside the limits. A proper value for the maximum forces can be obtained for each analysis from the tether characteristic, but that is out of the scope of this thesis. An example of this plot obtained from *Kite_Physic* shown in figure 33.

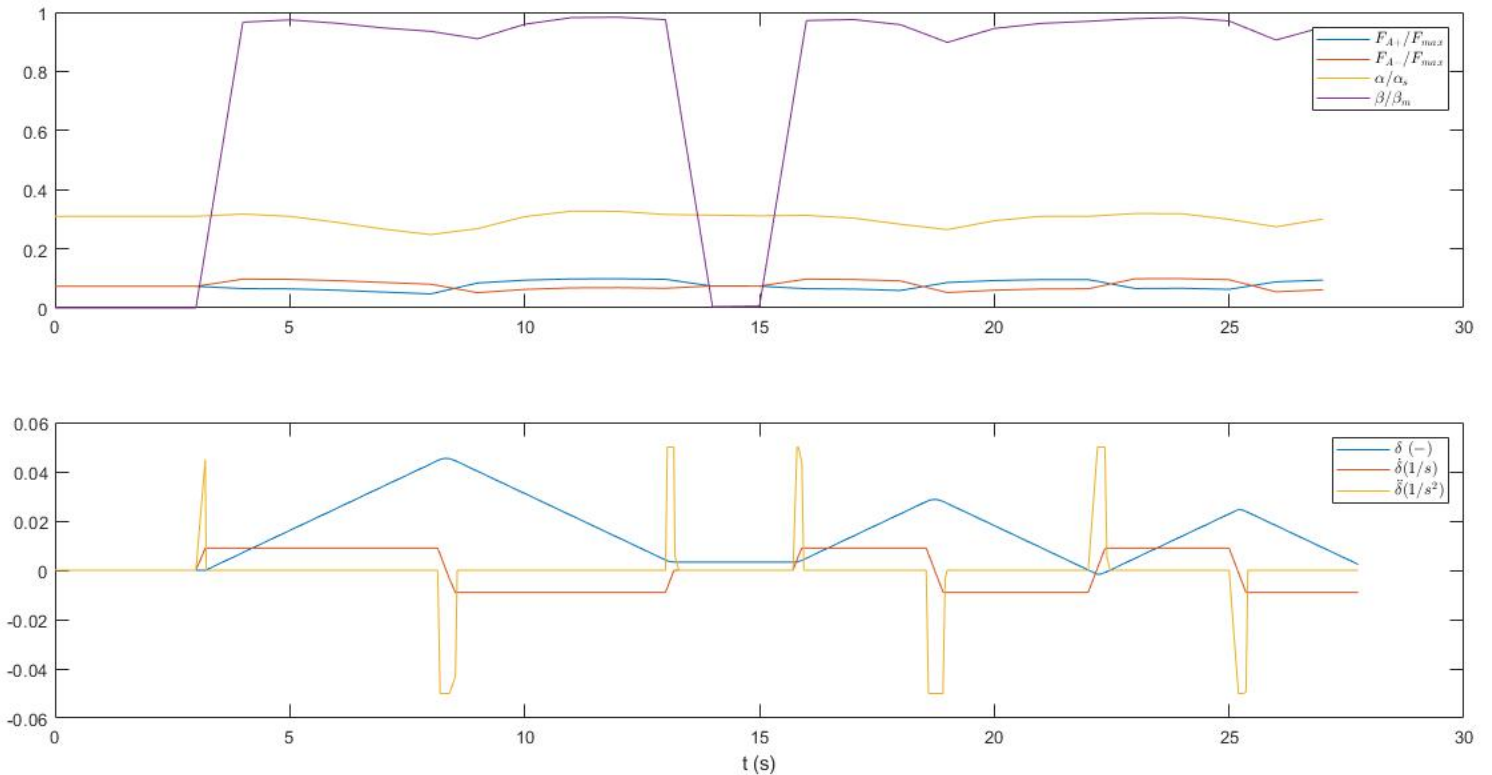


Figure 33: Kite physic plot [11]

In the control law model, limit values for δ and its derivatives must be settled. Thanks to the creation of the code `Kite_Physic.m`, it can be analyzed the influence of these values in the kite behaviour, in particular check for what values, the model is outside of the physical boundaries. Different model are created, varying those values and analyzing with the tool `Kite_Physic`. The different configuration are shown in table 3, and the result for each case can be observed in figures 34, 35, 36 and 37:

	Case 1	Case 2	Case 3	Case 4
δ_{max} (rad)	0.1	0.1	0.1	0.1
$\dot{\delta}_{max}$ (rad/s)	0.005	0.05	0.01	0.01
$\ddot{\delta}_{max}$ (rad/s ²)	0.05	0.05	0.05	0.5

Table 3: Maximum values configuration

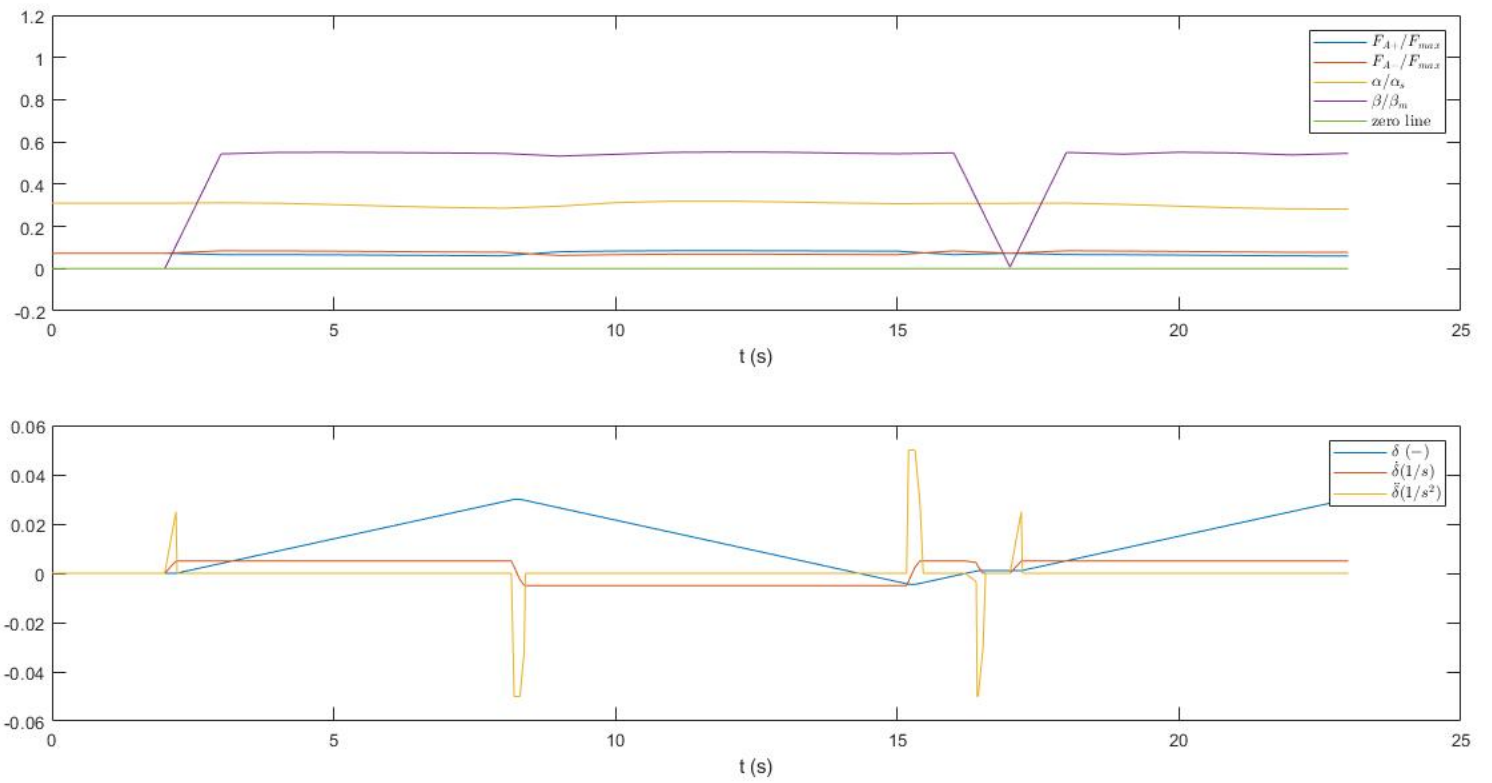


Figure 34: Case 1 [11]

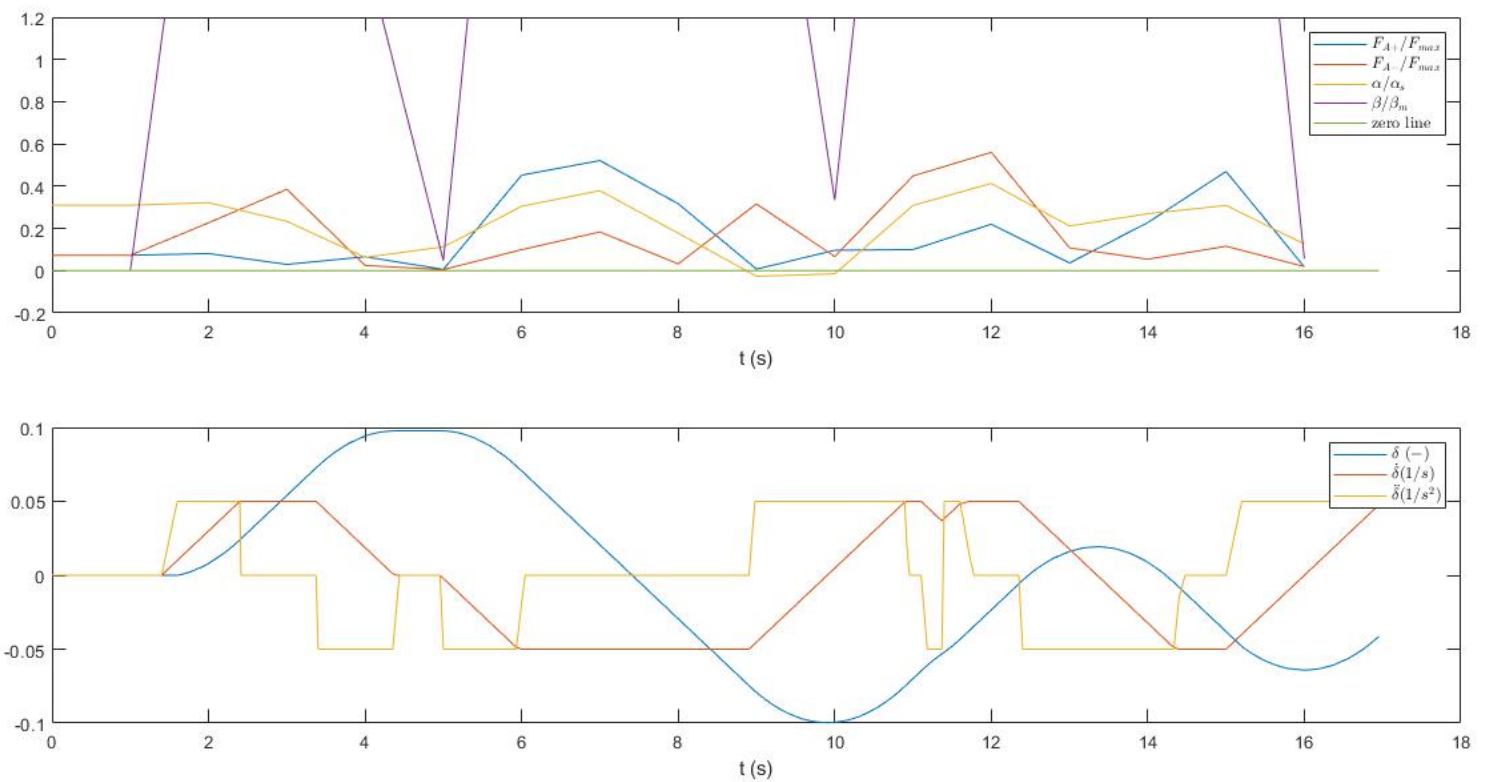


Figure 35: Case 2 [11]

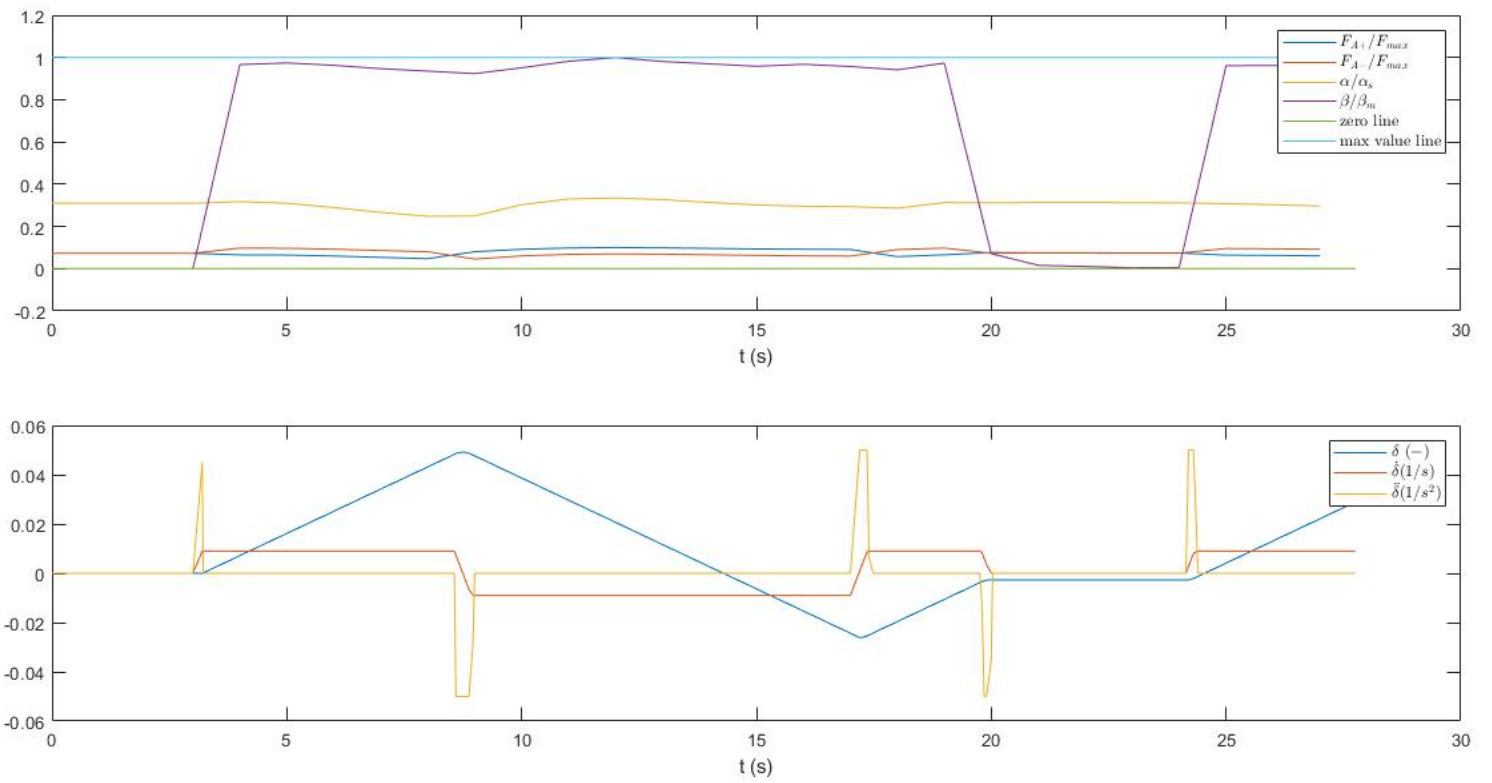


Figure 36: Case 3 [11]

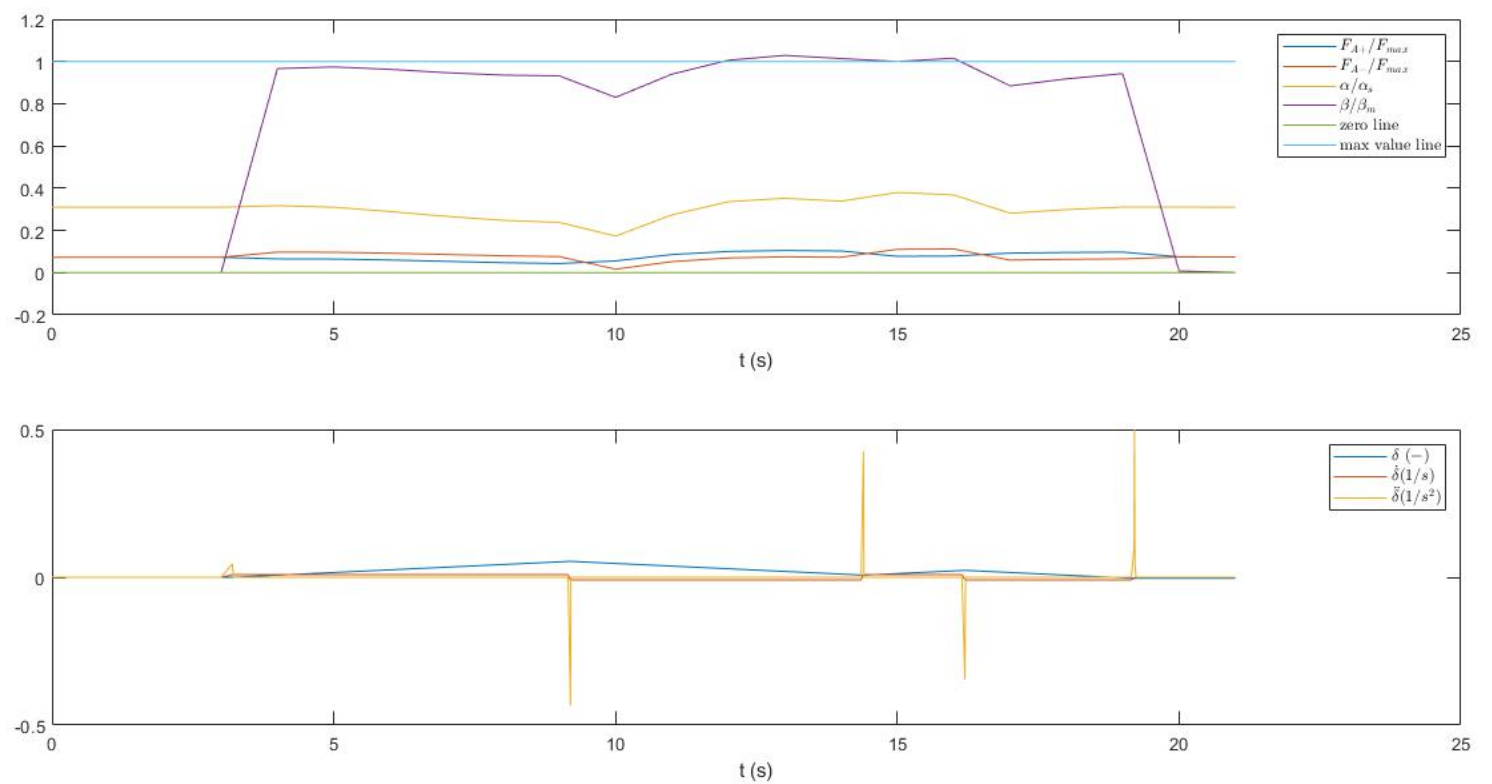


Figure 37: Case 4 [11]

From this analysis some conclusion can be obtained: β is the values most influenced by the user input, while the other remain mostly unaltered. $\dot{\delta}$ is the value that has main influence in β , as it can be observed, between case 1, 2 and 3, bigger its value, bigger β . Comparing case 3 and 4, it can be observed that $\ddot{\delta}$ has influence in β , but it is not such heavy as $\dot{\delta}$. From this analysis it can be conclude that $\dot{\delta}$ and $\ddot{\delta}$ must be kept inside some defined limit, and in the case of this thesis they are $|\delta| < 0.1 \text{ rad}$, $|\dot{\delta}| < 0.01 \text{ rad/s}$ and $|\ddot{\delta}| < 0.05 \text{ rad/s}^2$. The value of δ is limited to the change on the length of the tether. In this case, the tether is operated by an human being, so it must be limited due to human physiology.

Calculation of maximum value of δ

This thesis simulate an acrobatic kite, which are operated by a person. The person controlling the kite, has influence in the values of L_+ , and L_- , the longitude of the tethers. Due to human physiology (the length of the arms), it has been assumed that the tethers length can change $\Delta L = \pm 25 \text{ cm}$ per tether. Making calculations, the value of δ , in the case of this thesis, for the most extreme cases of $L_+ = L_0 + 0.25$ and $L_- = L_0 - 0.25$, being $L_0 = 200 \text{ m}$, is: $\delta = 0.1$. This calculation has been made using formulas (8) and (9), explained in section 3. For the purpose of this calculation, a code has been created, in which the user set the maximum value of variation of the length, the length L_0 , and the program output the maximum value of delta. This code will be useful in the future, as it can be used to calculate this value in other kite models, f.e. the one that the tether length is controlled by a machine; in which case, the length variation is different.

4.7 Task influence in real time synchronization

The block "Real Time Simulation" has an option called missed ticks, which provide important information, regarding the performance in real time of the simulator. The user has influence in some parameters that influence the computational time, and thus the real time synchronization, which are the following:

- Rate of plot of the kite: The block "Transition Rate", allows to plot the kite at a rate different that the one of the simulation. In figures 38, 39 and 40 different cases are plotted. It can be observed that for rate of 0.5s, simulation at real time is not achieved and for rates greater than 1s, the one chosen in the program, it is. It must be notice that the program demand computation resources in other tasks, like the δ input and that is why one task, like the display, can not use all the computational resources. Another interesting conclusion is that every time the plot is made, the missed tick increase to 1, and at the beginning of the simulation, the missed ticks are 2. This is because the plot is the most computational demanding task.

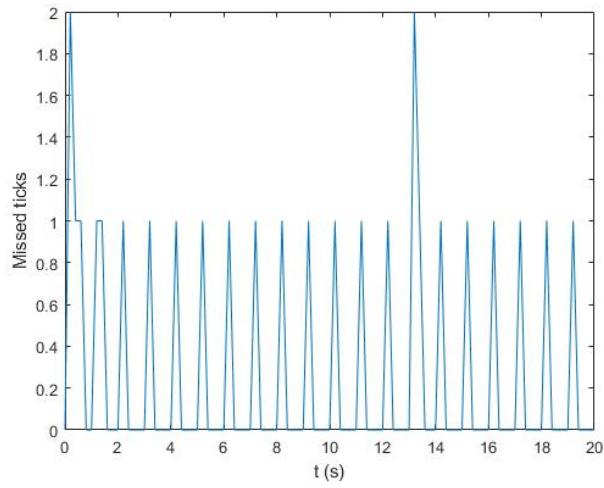


Figure 38: Plot rate t=1s

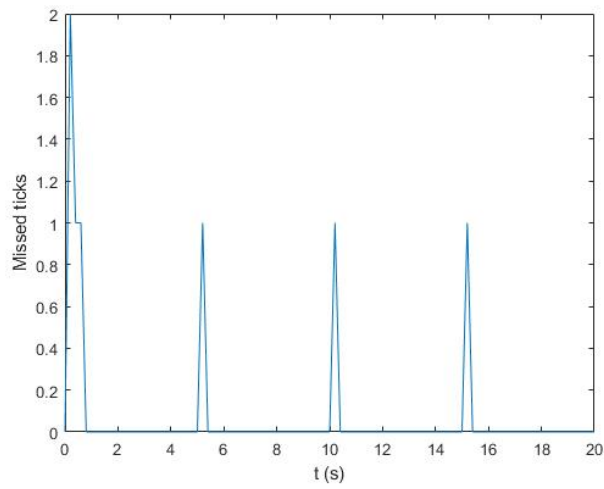


Figure 39: Plot rate t=5s



Figure 40: Plot rate $t=0.5s$

- The graph initialization is computational time demanding: It is recommended to initialize them before performing a simulation. In figure 41, and 42 missed ticks as function of the time are plotted for two cases: when figures and axis are not initialized and when they are, all without any control law acting on the kite. It can be observed that in the first case, the program require 10 seconds to achieve the real time simulation, while in the second one it start to simulate at real time.

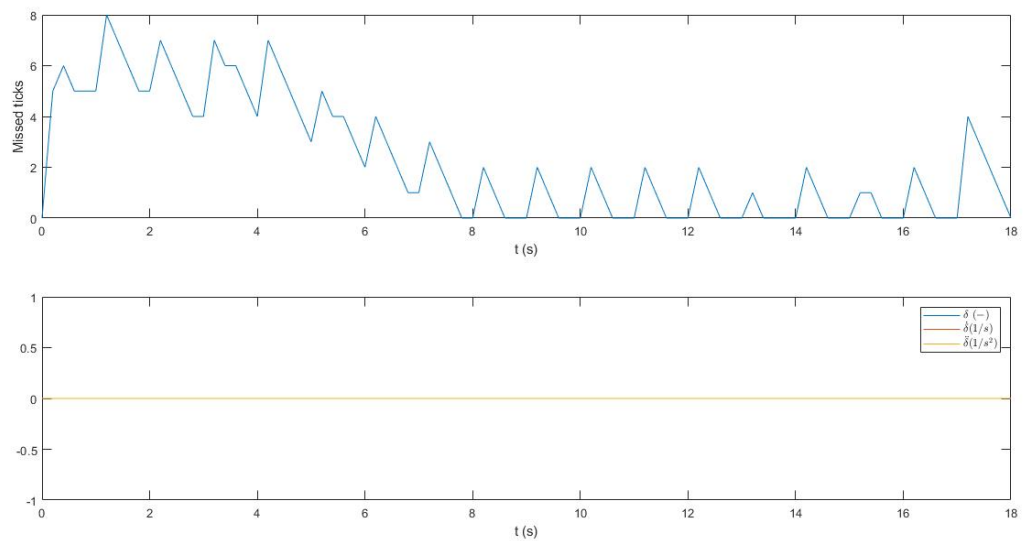


Figure 41: Figure not initialized [10]

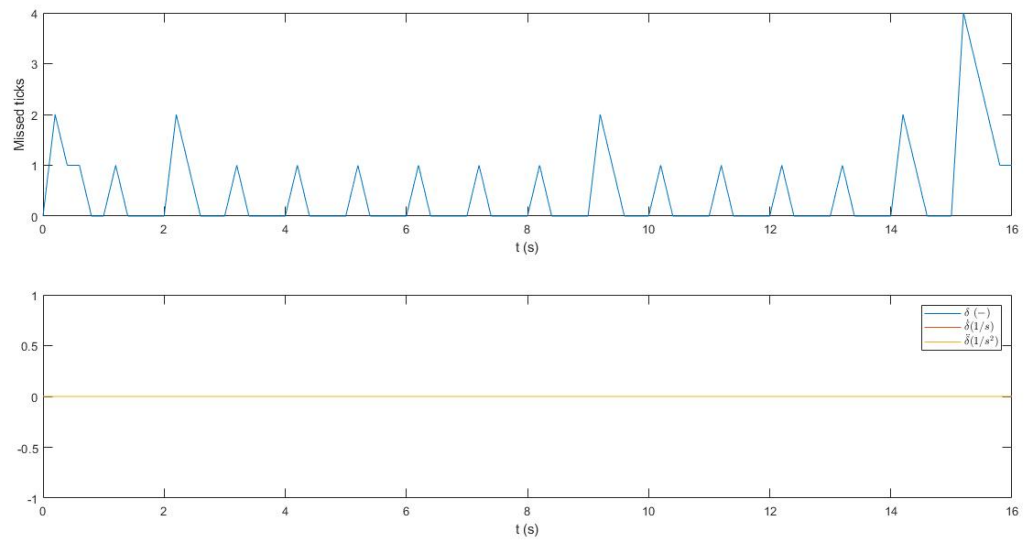


Figure 42: Figure initialized [10]

- Influence of the frequency of change of the control input: The program is modified to study the influence of the frequency of change of the control vector in real time simulation. The constant is substituted for a Matlab function that changes the value of δ from 0.15 to -0.15 at a specific rate. It can be observed that the missed ticks are negligible for the cases $t=5$ s, $t=1$ s, $t=0.4$ s, but when the frequency is so small ($t=0.1$ s) missed ticks increase in every time step and real time simulation cannot be achieved. The different results can be observed in figures 43, 44, 45 and 46.

4 SIMULINK PROGRAM

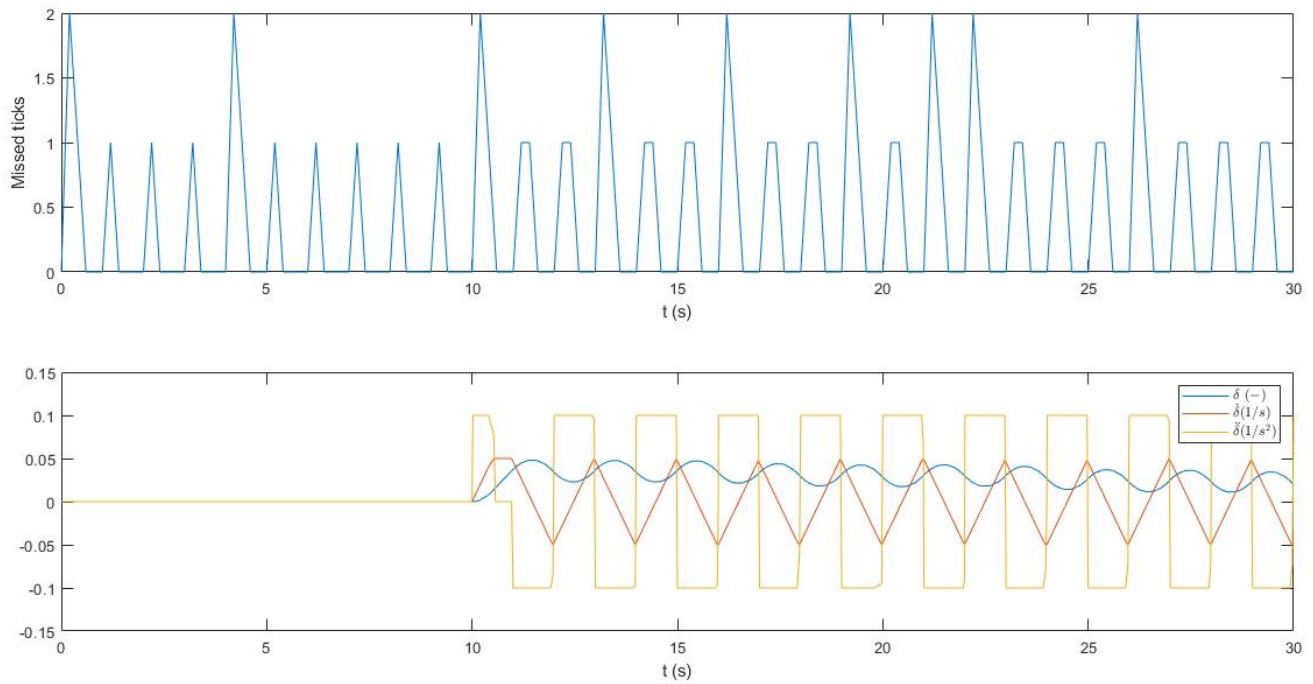


Figure 43: Case $t=1s$

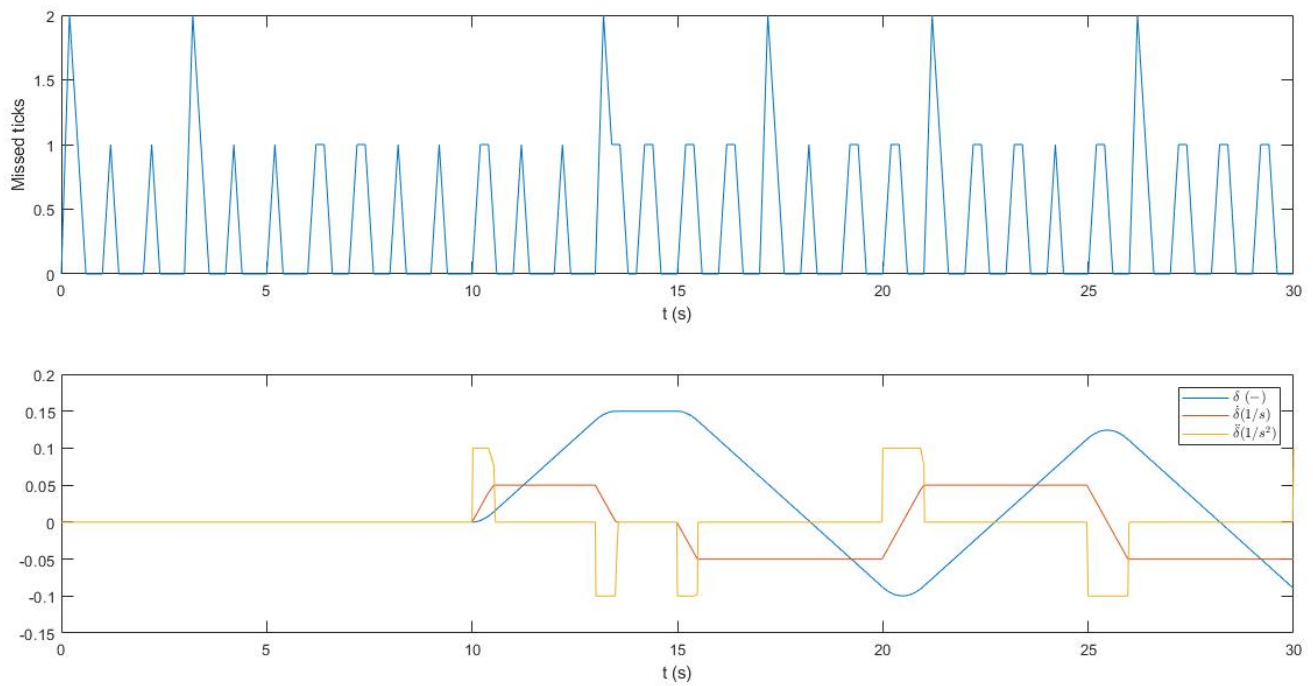


Figure 44: Case $t=5s$

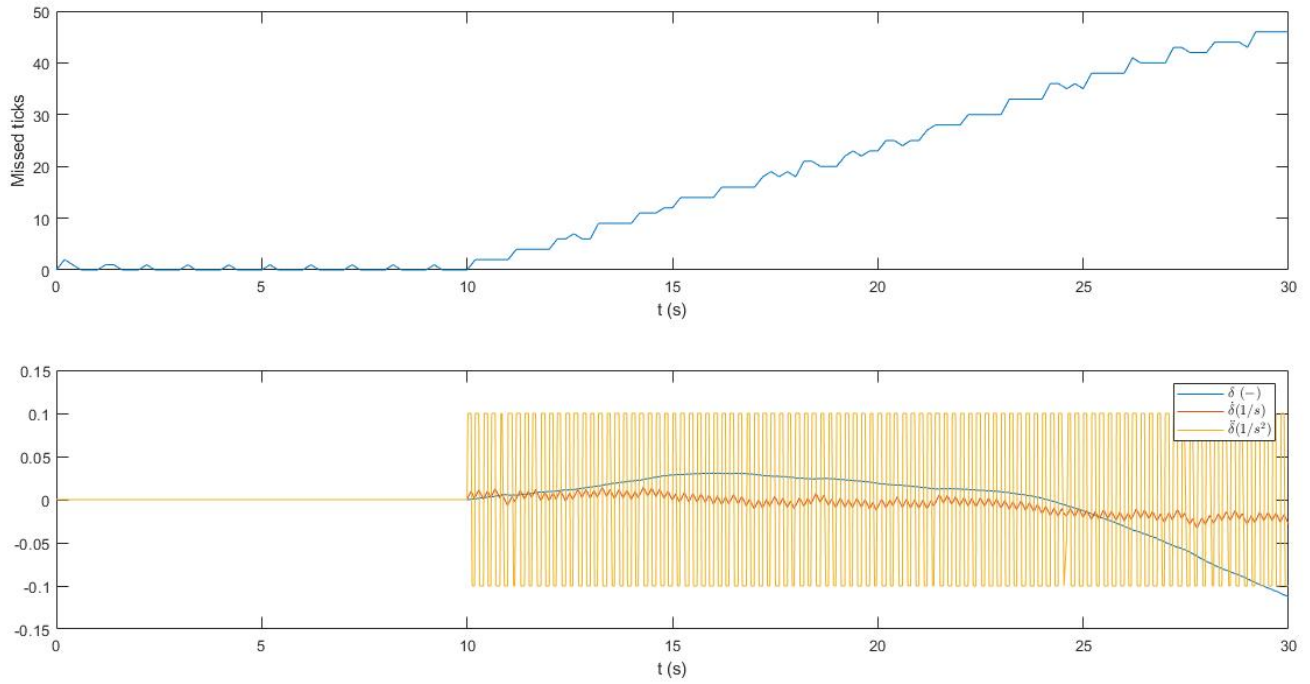


Figure 45: Case $t=0.1s$

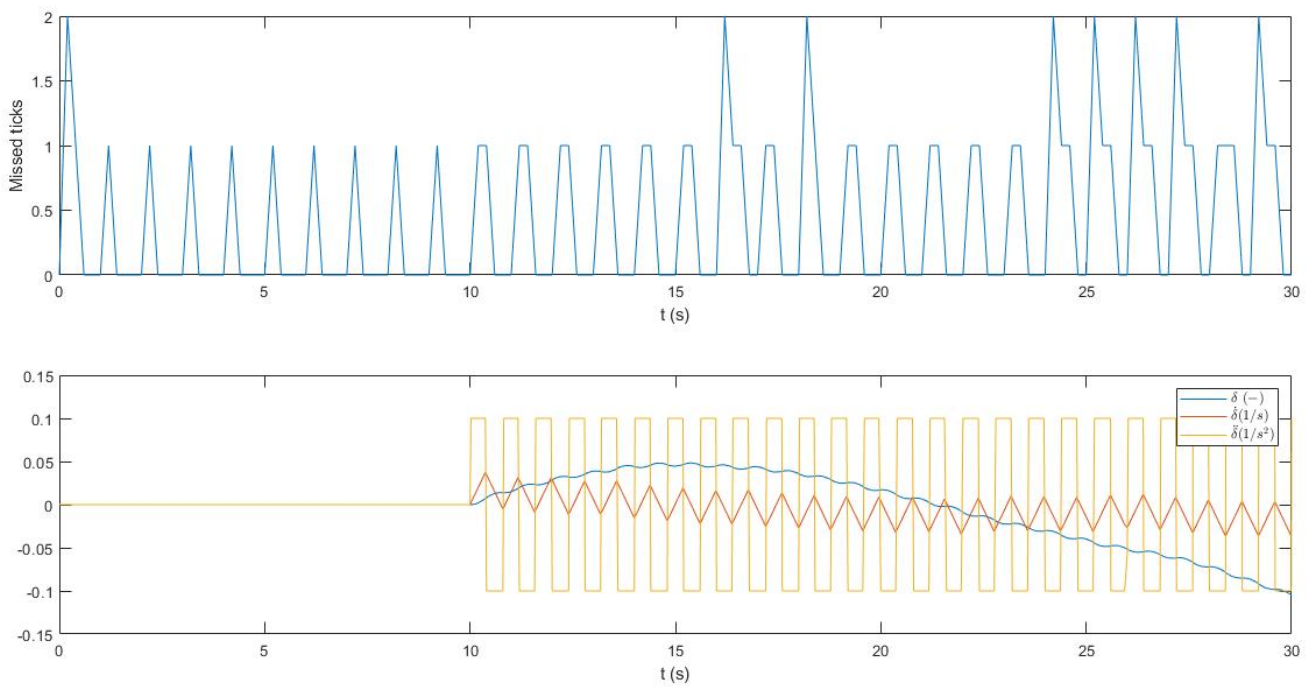


Figure 46: Case $t=0.4s$

5 Regulatory Framework

This thesis is about the creation of a simulator, that is why there are not safety constrain or regulation that affect it, only some issues regarding author rights must be taken into consideration. LAKSA is a freely available software, it can be used, although it has all right are reserved. The whole program and the post processing of the data have been developed under Matlab and Simulink standard programming. In order to have a legible program, and for an easy understanding of it for further development, it must follow good programming practices, that has constrain the development procedure:

- Structure: proper structured with an structure with sense and easy of understand.
- Modular program: program must be subdivided in functions, each one with a defined task.
- Comments in the code: comment are essential to understand what is done in each part.
- Self-explanatory function titles: the title must describe function task.
- Simplicity: Avoid unnecessary computations and blocks, when programming in Simulink.
- Variables: Proper explanation of the variables used and its units.

Regarding the use of a kite, although it does not affect directly this thesis, it is important to consider the regulatory situation of these flying devices. Nowadays there is an increase in the users of air, f.e. the amount pilot of drone is increasing, due to its easy and cheap access. All these activities must be regulated in order to allow every user of the air use it in a safety way while the air space is used efficiently. Nowadays in Spain, big kites, and kites that fly at a high altitude are regulated and in order to fly them, permissions are required. In the development of bigger kites which can fly higher, this problem will even increase, and it must be regulated in order to avoid interference with aircraft. In addition, safety condition and safety operations must be ensured under all circumstances. This software is a good chance to test extreme cases in a safety way. Regarding the use of information and pictures in this thesis, they come from a free-access source, and they have been properly reference, in order to accomplish with the law and the intellectual property.

6 Budget

In this section, the total cost of this project is going to be explained. This project is based on the creation of a program, and for that reason, for its development it is only needed workforce, hardware and software. Regarding the workload, it consists on a Junior and a Senior engineering. The hardware is composed by the computer used for the development, a "HP Pavilion 15 Notebook PC" and the software is based on Matlab and Simulink licensing. The Junior Engineering work half time (4 hours per day, 5 days a week) during 3 month, which is the time spend in the development of the program. It is assume that the Senior Engineering work 35 hours, about 14 %, see table 4. Regarding Simulink and Matlab price, although in this thesis the university license is used, the real price is added in order to be closer to the reality. It is assume that the program is used during the whole year, and during the development of the project, which last 3 months, it is exclusively used for this project. For the hardware, the useful life is assumed to 5 years, see table 5. Adding all this prices, the total cost is calculated and it is: 5193.8 €, see table 6.

	Salary (€/h)	Time (h)	Price (€)
Junior Engineering	15	240	3600
Senior Engineering	30	35	1050

Table 4: Employees cost

	Duration (month)	Adquisition cost (€)	Time used (month)	Cost (€)
HP Pavilion 15 Notebook PC	60	876 [13]	3	43.8
Matlab	12	800 [14]	3	200
Simulink	12	1200 [14]	3	300

Table 5: Tool prices

Item	Cost (€)
Junior engineering	3600
Senior engineering	1050
Computer	43.8
Matlab	200
Simulink	300
Total cost	5193.8

Table 6: Total cost

7 Conclusion

Project conclusions

At the end of all of this implementation process, the final model is obtained. This model has some strengths, like its modular behavior, thanks to the independency of each part, making each one easy to improve. The improvement made in Simulink with respect to LAKSA program, is that real time simulation is achieved; the control of the kite is done at real time, in such a way that it does not need to follow a function, and the display is done at the same time as the calculations. Three different parts can be easily recognized: user input, equation solver and kite display. This type of simulator and studies are good to know the application and viability of kites, at a very low cost, taking advantage of the data obtained from the simulator. Thanks to its structure this model can be applicable to other types of kite, in particular for kites linked to the Earth, so different models can be analyzed easily; and the capabilities of different kites studied. This project has proved that with this kind of equations, the program can simulate in real time. The influence of different control laws has been studied too, and an easy way of checking the physical consistency of the model has been developed. Some interesting data regarding the performance of the program in real time is obtained, which can be used in future projects in order to improve the performance of the program.

Further projects

Although this model has made a big improvement with respect to its predecessor, LAKSA code, some parts can be improved and can lead to future theses. Moreover this program can be the key to make other studies:

- Control input: It is desired to substitute the knob with a joystick, as it has more functionalities, and full control of the kite could be achieved (l and δ). At the beginning of this thesis an attempt of using the joystick was made, but this solution was rejected due to license problems.
- Autonomously control: A function which controls the kite autonomously can be developed. This function can read values of the actual state of the kite, and control the kite accordingly.
- Improve the aerodynamic model: The aerodynamics of the kite is not deeply developed, and a better model will provide results closer to reality.
- Display of the kite: In this thesis the display of the kite is done using MATLAB plot function. It has some advantages, like using LAKSA function, but Simulink offers other solutions which require less computational time, like Simulink 3D animations. This solution has been considered, but rejected due to license problems.

References

- [1] G. Sánchez-Arriagaa, M. García-Villalba,*, R. Schmehl.. *Modeling and dynamics of a two-line kite*. Applied Mathematical Modelling 47, 2017, 473–486
- [2] Makani Technologies LLC.
<https://makanipower.com/>
- [3] awec2017
<http://awec2017.com/>
- [4] AWEC 2017 book of abstracts
<http://awec2017.com/abstracts.html>
- [5] AWEC 2017 book of abstracts, p15, Progress and Challenges in Airborne Wind Energy.
<http://awec2017.com/abstracts.html>
- [6] Ampyx Power B.V.
<https://www.ampyxpower.com/>
- [7] SkySails
<https://www.skysails.info/>
- [8] G. Sánchez-Arriagaa, M. García-Villalba,*, R. Schmehl.. *LAKSA simulator* Matlab, 2017
- [9] G. Sánchez-Arriagaa, Alberto José García Muñoz. *Simulink program* 2019
- [10] G. Sánchez-Arriagaa, Alberto José García Muñoz. *Post processing LAKSA data* Matlab, 2019
- [11] G. Sánchez-Arriagaa, Alberto José García Muñoz. *Kite_Physic* Matlab, 2019
- [12] J. Alonso-Pardo, G. Sánchez-Arriaga *Kite model with bridle control for wind-power generation* J. Aircraft 52 (3) (2015) 917–923, doi:10.2514/1.C033283
- [13] PC componentes web page
<https://www.pccomponentes.com>
- [14] Matlab web page
<https://es.mathworks.com/pricing-licensing.html>

8 Appendix A: LAKSA program and function

This thesis is based in a Matlab code, called LAKSA, which simulate analytically the behaviour of a kite. The understanding of this code is crucial for a good and efficient development of the Simulink program, the goal of the thesis.

Main script

The matlab code is composed of a main script and functions. Looking at the main script an overview of the simulation process can be known. Deep calculations are developed inside the functions.

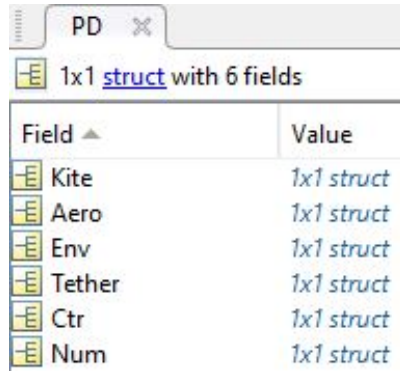
The main script can be divided in 4 parts:

- Creation of general variables. PD, PND, u0 and "options" are created.
- ode45 solver. The state vector is obtained at every time step using ode45 solver. The function Fun_ODE_Lag is crucial in this step, because it create $d\mathbf{u}/dt$ as function of simulation and input parameters, and $d\mathbf{u}/dt$ is used to calculate \mathbf{u} .
- Kite display and data analysis. Knowing \mathbf{u} at every time step, geometrical parameters of the kite are calculated, (like cords length...). This parameters are used to plot the kite and make some graphs regarding kite behaviour.
- Calculation are repeated with Hamiltonian formulation. This kind of calculation is not used in Simulink, so it is not going to be explained.

In the following sections, matlab functions are going to be explained.

Fun_PD_KA

This is a function which output an struct with all the physical parameters of the Kite in dimensional form (it has no input). Here are established different parameters divided in these categories (see 47): PD.Kite, PD.Aero, PD.Env, PD.Tether and PD.Ctr. PD.Kite, include geometrical parameter such as (chord, mass, span, Ix, Iy, Iz, Ixz). PD.Aero, is composed of all the aerodynamic data, like force coefficient (Cx0, Cxalfa...), torque coefficient (Clbeta, Clp, Cm0, Cmqa...) and limits of the model, stall angle (α_s) and maximum slide slip angle (β_{max}). In PD.Env all the enviromental values can be obtained (gravity acceleration, air density...). In PD.Tether the values that describe the tether can be founded (L_0 , X_A ...). PD.Ctr is composed of values related with the control of the kite, but in this thesis are not used, as the control does not follow a predefined function. In PD.Num is composed of all the values regarding the simulation set up (tolerances, iterations...). In tables 7, 8, 9 and 10, it can be found the most relevant values.



Field	Value
Kite	1x1 struct
Aero	1x1 struct
Env	1x1 struct
Tether	1x1 struct
Ctr	1x1 struct
Num	1x1 struct

Figure 47: PD struct [8], [9]

Chord (c)	1.5 m	Span (b)	5.8 m
Mass (m)	4 kg	Surface (A)	14.4 m ²
I_x	21.1 kg/m ²	I_y	4.66 kg/m ²
I_z	18 kg/m ²	I_{xz}	0 kg/m ²

Table 7: PD Kite [12]

α_s	25°	β_{max}	15°
C_{x0}	-0.065	$C_{x\alpha}$	0.176
$C_{l\beta}$	-0.1	C_{lp}	-0.15
$C_{y\beta}$	-1.57	C_{m0}	0.1332
$C_{m\alpha}$	-0.7633	m_q	-0.165
C_{z0}	0.116	$C_{z\alpha}$	-2.97
$C_{\eta\beta}$	-0.027	$C_{\eta r}$	-0.002

Table 8: PD Aero [12]

Earch acceleration (g)	9.81 m/s ²	ρ	1.225 kg/m ³
Wind velocity	7 m/s	Tether length (L_0)	200m

Table 9: PD Env [12]

L_0	200 m	X_A	0.75 m
Y_A	2.9 m	Z_A	2m

Table 10: PD Tether [12]

Fun_PND_KA

This function change PD values from dimensional to non dimensional form. Its input is PD and its output is PND, a struct variable with the same configuration as PD. The values of PND are shown in table 11. In order to change from dimensional to non dimensional quantities the following values are used:

- $L_0 = 200$ m
- $g = 9.81$ m/s^2
- $m = 4$ kg

Symbol	Value	Symbol	Value	Symbol	Value
ϵ_c	0.0075	ϵ_b	0.029	ν_W	0.25
x_A	0.0037	y_A	0.0037	z_A	0.01
i_x	$1.32 \cdot 10^{-4}$	i_y	$2.92 \cdot 10^{-5}$	i_z	$1.12 \cdot 10^{-4}$
μ	440.4	β_{max}	15°	α_s	25°
C_{x0}	-0.065	$C_{x\alpha}$	0.176	$C_{l\beta}$	-0.1
C_{lp}	-0.15	$C_{y\beta}$	-1.57	C_{m0}	0.1332
$C_{m\alpha}$	-0.7633	m_q	-0.165	C_{z0}	0.116
$C_{z\alpha}$	-2.97	$C_{\eta\beta}$	-0.027	$C_{\eta r}$	-0.002

Table 11: Dimensionless parameters used in the simulation [12]

Fun_ODE_Lag_KA

This function has as input t and $\mathbf{u}(t-1)$ and output $d\mathbf{u}/dt$. t is used to create the control vector (it is as function of time), using Fun_Control_KA; and wind velocity, using Fun_Wind. $\mathbf{u}(t-1)$ is used to create the rotation matrix and getting values of different kite components. After some calculation $d\mathbf{u}/dt$ is obtained. This values are so important, because they are integrated to calculate \mathbf{u} .

Fun_Control_KA

This function create the control vectors, analytically, as function of time and PND. In the original code there are three types of control:

- no control (l is kept constant and $\delta=0$ and constant)
- δ change (and l is kept constant). In particular the change is delta is: $\delta(\tau) = \delta_0 \cdot \sin(\omega_\delta \tau)$
- l and δ change.

This function will be modified in the future, and the control vector will be created using the user input.

Equilibrium_KA

This function calculate the (initial) equilibrium position of the kite and check if the position calculated is a real equilibrium position, if it is not, it will give an error.

Fun_Post_KA

This function read values of PD, PND, t and \mathbf{u} , and calculate values of the kite component, theder tension, aerodynamic forces, euler angles... When the data of Simulink is analyzed,

this function is used, and in this case it must be modified to read the values of the control vector. This function calculate kinematic, forces and moment components of the kite, see tables 12, 13 and 14.

RBE	Body-Earth rotation matrix
rk	Earth components of the position
vk/ak	velocity and acceleration vectors of the center of G
euler	Euler angles
omega	Earth components of the kite
omega_p	angular velocity and acceleration

Table 12: Kitematic component

Lambda	Tether Tensions along tether directions
FAP	Earth component of the Force exerted by the tether linked at A+
FAM	Earth component of the Force exerted by the tether linked at A-
MAP	Earth component of the Torque exerted by the tether linked at A+
MAM	Earth component of the Torque exerted by the tether linked at A-
FA	Earth component of the Aerodynamic force
MA	Earth component of the Aerodynamic torque
W	Earth component of the kite weight

Table 13: Kite forces and moments

alpha	Angle of attack
beta	Sideslip angle
LP	Length of the A+ tether
LM	Length of the A- tether

Table 14: Other components

This variables are used to make different calculations when the result are obtained.

Plot_KA

This function or display the whole kite. First it set the axis and the figures. This is an important point to take care of, because in the Matlab code the axis are set every time the function is call, making the simulation slow. This will be improved to achieve a good computational time. The kite is display in two windows, one showing the whole Kite, and other focus on the cloth (the last one is called "plot insect of the kite"). The tether is plot in the function itself and the cloth is plot using Plot_Kite. The final plot of the kite can be observed in figure 48.

Plot_Kite

This function plot the cloth of the kite. It has as input the center of mass, span, chord and height of the kite.

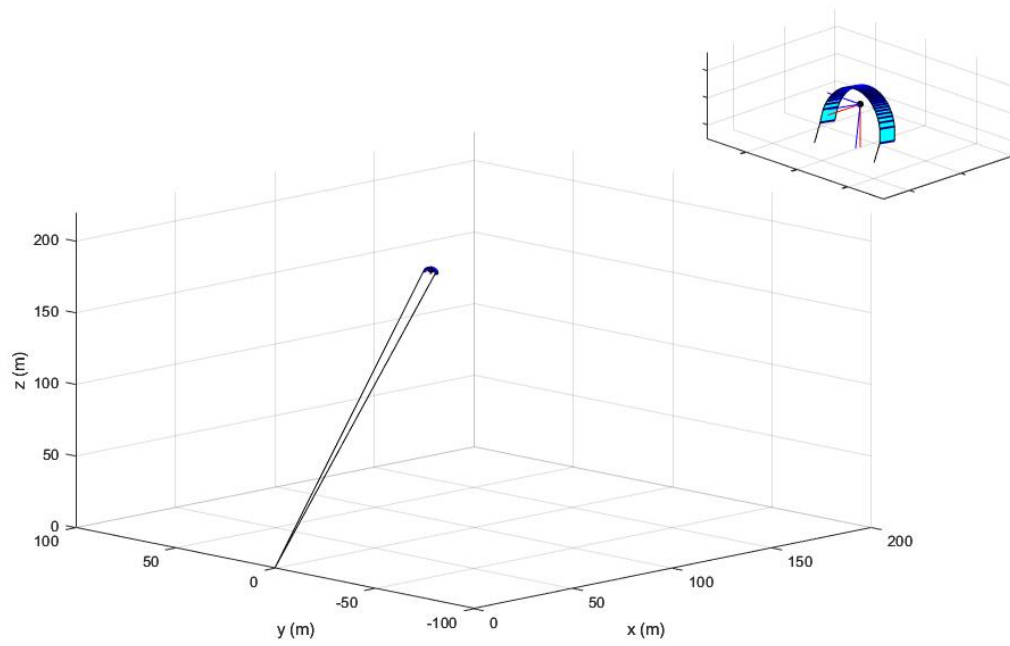


Figure 48: LAKSA kite display [8]

9 Appendix B: Simulink Blocks

The Simulink code is developed using different block. Understand them is very important to understand how the program works and the decision made.

- Interpreted matlab function:

This block read Matlab functions. In the case of this thesis it has a lot of advantages as it allows to use the original code without big modification. One of the problem faced at the beginning of the development was how to deal with struct variables, as in Simulink can not be used. If a one signal in the Simulink program is a struct, it leads to errors, which is a problem because the original code is full of this kind of variables. The advantage of work with "Interpreted matlab function" is that it allows to use the original functions, while working with struc variable. The main disadvantage is that the block has only one input and one output port. In case that multiple variables as input or output they must be separated or mixed, as required, and the function must be modify to read only one variable too. In case that no inputs our output are needed, the program must input or output a value, although it will not be used. In the output case the signal must finish in a terminator block. "Interpreted Matlab Function" block is shown in figure 49.



Figure 49: Interpreted Matlab Fcn [9]

- Mux and demux:

The Mux block combines its inputs into a single vector output. An input can be a scalar or vector signal. All inputs must be of the same data type and numeric type. The input can be a vector or a scalar, but to simplify the understanding of the signal flow, in this thesis, only scalar are input. The demux block works in the other way around than the mux block. Its input is a vector, and its output are the different vector components (vectors 1x1). In order to avoid errors, the vectors must have the same size during the whole simulation. Both blocks are shown in figure 50.

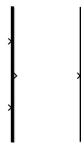


Figure 50: Mux and demux [9]

- Constant:

The Constant block generate a signal with some defined value. The value can be a scalar, vector or matrix and it can be defined in different ways: the box has the option to write the value (in this case it will be constant during the whole simulation), associate the value to a knob (in this case, it can change during the simulation and not be constant anymore). The box, connected to a knob can be observed in figure 51.

- Knob:

The Knob block tunes the value of the connected block parameter to during simulation. In this thesis, the connected box is a constant block.

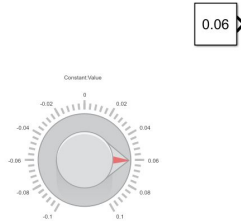


Figure 51: Knob connected to a constant [9]

- Integral:

Integrator block integrate the input signal with respect to time. The solution of the integral is the output. The initial value must be establish, and there are two option for doing it: it can be write inside the box or it can come from an external signal. This is an example of how its works: given a function $\dot{x}(t) = u(t)$ as input, for an initial condition $x(t_0) = x_0$, the output will be $x(t)$. The block can be observed in figure 52

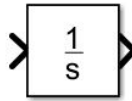


Figure 52: Integral block [9]

- Derivative

The derivative block calculate numerically the derivative of the input as function of time (which is the output). Given a function input u , $\frac{du}{dt}$ is obtained by computing $\frac{\Delta u}{\Delta t}$.

- Rate transition:

Rate Transition block allows to transfer the input data at a defined time step. In this thesis is so useful, because it allows to operate at two different rate: one rate in calculation part and another rate in the display part.

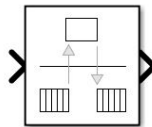


Figure 53: Rate transition Block [9]

- Terminator:
Terminator block allows to finish an unconnected signal. This block is very useful when there is no output value from "Interpreted Matlab Function"

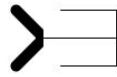


Figure 54: Terminator [9]

- to_file:
To file block is used to save a signal into a .mat variable, which can be a scalar or a vector and in addition, it saves the time at the different simulation steps. In this thesis, it is used to save data in Array format.



Figure 55: To file box [9]