Università degli Studi di Padova

Dipartimento Di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria delle Telecomunicazioni

Tesi di Laurea

Flocking of UAVs
Software model and limited vision simulations

Relatore: Prof. GuidoMaria Cortelazzo
Correlatore: Prof. Adrian F. Clark
Correlatore: Prof. John C. Woods

Candidato: De Nardi Renzo

Lavoro svolto presso il VASE lab, University of Essex (UK)

Anno Accademico 2003-2004

Contents

Sommario

La presente tesi è parte del progetto GRIDSWARM [1] sviluppato all' ESE Department della University Of Essex (UK), nel quale si punta alla realizzazione di uno stormo autonomo di aeromodelli.

Il controllo del volo in stormo sarà decentralizzato, svolto autonomamente da ogni aeromodello in base alle informazioni raccolte mediante telecamere e sistemi di posizionamento. Una wireless network collegherà i singoli computer degli aeromodelli in un parallel computer dalla elevata capacità computazionale.

Un esempio di impiego tra tutti, potrebbero essere i rilievi fotogrammetrici, nei quali l'elevata area controllabile e le capacità di calcolo del gridswarm, costituiranno veri e propri punti di forza.

Il presente lavoro è dedicato in particolare all'analisi delle possibilità di ottenere un volo in stormo basato principalmente sulle informazioni ottenute tramite una telecamera posta a bordo del velivolo.

Al fine di ottenere una fedele simulazione della reale situazione da controllare, è stato creato un modello matematico a sei gradi di libertà dell'aeromodello. Un pilota automatico basato sulla teoria classica dei controlli è stato quindi preposto al modello per permetterne il completo controllo.

Di entrambi i sistemi più che la mera descrizione si è cercato di trasmetterne i principi guida, al fine di aumentarne la fruibilità da parte degli altri membri del progetto. I due sistemi creati sono stati inclusi in una unica libreria software, da utilizzarsi nei futuri sviluppi.

Un algoritmo per il controllo del volo in stormo basato sulle ben note regole di separazione e coesione è stato in fine proposto.

L'efficacia dei modelli matematici del velivolo,del suo pilota automatico e la validità del metodo per il volo in stormo sono stati dimostrati mediante simulazioni ad hoc.

Tramite una serie di simulazioni si è in fine dimostrato come, definitone l'algoritmo, la profondità dello spazio visivo e l'angolo di massima visione influenzino il volo in stormo. In particolare si è dimostrato come garantendo un angolo del campo visivo di almeno 110° ed una profondità dello stesso di almeno 80 m, si riescano ad assicurare ottime prestazioni in più del 95% dei casi.

[1] Informazioni più dettagliate sul progetto possono essere reperite all'indirizzo web *http://gridswarms.essex.ac.uk*

Il presente lavoro è stato realizzato presso il Vision And Syntetic Enviroments laboratory della University of Essex, UK.
Ringrazio sentitamante il Prof. Adrian Clark e il Prof. John C. Woods per la disponibilità dimostrata.

Ai miei Angeli

Chapter 1

Introduction

Imagine a large group of small unmanned autonomous aerial vehicles that can fly with the agility of a flock of starlings in a city square at dusk. Imagine linking their inboard computers together across a short-range, high-bandwidth wireless network and configuring them to form an enormous distributed parallel computer. Imagine using this huge computational resource to process the sensory data gathered by the swarm, and to direct its collective actions. You have now grasped the idea of a flying gridswarm. At Essex, we are working to bring this vision to reality.[1]

This is certainly the best way to introduce the gridswarm project, a project currently under development at the University of Essex; it is therefore also the best way to introduce this thesis, which is a part of it.

Within the interests of the project there are also heterogeneous swarms. The possibility to exploit a combination of UAVs and terrestrial robots, enlarge even more the variety of task in which the gridswarm could be employed.

For example this will allow, to use the privileged point of view of the airborne swarm to organise the action of the terrestrial vehicles. All the informations gathered from the terrestrial and airborne vehicles, will be eventually easily combined and processed using the large computational power of the swarm.

Moreover than its futuristic aim, the gridswarm projects is an opportunity to face a series of exciting challenges in different engineering subjects.

Unmanned autonomous flying vehicles

Since many years ago, the unmanned aerial vehicles are well known and used as radio-controlled models by amateurs. In the recent years the advancement in miniaturisation of electronics, and the introduction of Global Positioning systems, changed the way in which the UAVs are conceived. From simple leisure toys, the UAVs became proper vehicles.

The market firstly interested to the UAVs was as obviously the military one; the possibility to undertake critical mission without the employment of human pilot is of course interesting.

The cheap and sophisticated inboard autonomous autopilot systems available nowadays [3] as a matter of fact lead the UAVs also to the commercial market.

[1] The gridswarm vision quoted from the gridswarm website *http://gridwarms.essex.ac.uk*

Swarm technologies

The theory and practice of the distributed control of swarms of agents was introduced in 1987 by Craig Reynolds[2]. The essence of his findings was that excellent coordinated group motion can be produced only with a distributed control. Each agent has to move according to a small set of simple rules, that take into account the range bearing, and preferably the orientation, of its neighbours.

The decentralised control of the vehicles has some undeniable advantages. As first it is highly scalable, no difference there are between controlling 10 or 100 vehicles. Secondly, no fixed formations are predefined; higher flexibility and higher capacity of manouvre can therefore be exhibited. Thirdly, the type of control is intrinsically fault tolerant; the failure of an agent does not compromise the flock.

Other authors presented variants of these rules producing real-time flocking in real robots; two examples are the work of Ian Kelly [4] and Adam Hayes [5].

Cluster computing

The idea behind the gridswarm had its origins in the Beowulf Project. The Beowulf Project demonstrated the possibly of creating a distributed parallel computer interconnecting through an Ethernet network a number of cheap Linux boxes. The following step was done by Owen Holland and Alan Winfieldwith in the 1999 with the LinuxBots; a groups of small autonomous mobile robots running Linux and equipped with `IEEE 802.11b` wireless LANs. Alex Holland, remarked that it should be possible to configure a fleet of LinuxBots to operate across the wireless LAN as a Beowulf cluster. A swarm of robots, forming a free parallel distributed supercomputer... the gridswarm concept was born.

Unfortunately at that time the potential of the idea was limited by technological issues. Conversely, nowadays the technology is mature to realize the idea.

In the recent years technical improvements in the wireless network technology (e.g., IEEE 802.11a, HYPERLAN/2) and in the mobile platforms, (faster and cheaper low power processors) gave the basis for a successful development.

Interestingly,the main cluster computing organisation[6] does not seem yet to include wireless interconnects among its network technologies, one day soon probably it will.

The brief survey of systems for mobile computation[7] of Stelios Buonanos is an interesting overview on the state of art technologies for mobile computation.

The platform

Our initial experimental platform is a Chris Foss WOT4 Classic, its wing span is 52" and it is equipped with a 0.75 4-stroke motor (see fig. 1.1). The large wingspan and its classic high wing design ensure a superb manoeuvrability, and exceptional low cruise speeds; for this reason was preferred to other aircraft models. So far the WOT4 was used principally for testing the equipments. Test were run on the autopilot system (a MicroPilot MP1100), on the communications loop, on-board cameras, and range-and-bearing systems. The large dimensions of the WOT4 allow easily to integrate the autopilot and the others electronic equipments.

Figure 1.1: Picture of the WOT4

Faithful 3D virtual models of the WOT4 were also realized, they will be used on the development of the vision algorithm.

Similar projects

Several US projects are aimed at getting UAVs to fly in formation, usually under remote but high-level control. This type of projects are therefore different from the biologically-inspired flexibility and responsiveness of flocking pursued within gridswarm. However many of the required technologies are similar.

- Probably the project more similar to gridswarm is *The MinuteMan project* at UCLA[8]. Aim of the project is to build a reconfigurable architecture for highly mobile multi agent systems. In the intention the computationally-capable autonomous vehicle would be able to share information across a wireless fault tolerant network.

- Study on the formation-flying were undertaken at MIT, within the *Autonomous blimps*[9] project.

- Slightly different from the previous, *The flying flock* project developed at the University of West England [10]; here the work is conceived with a minimalist approach.

- Other organisations drive their effort into development of UAV platforms specifically designed for swarm systems or formation flying. An example is the *Dragonfly* project at Stanford[11].

Applications

Thinking about the gridswarm probably the first applications that pop up in anyone mind are the military ones however these are not the only.

Possible applications come from the field of environmental engineering, where a gridswarm would for example provide instantaneous air data monitoring over large atmospheric areas.

With the suitable equipment a gridswarm would be also usable for photogrammetry recording, or change detection in digital mapping. Detection of cropmarks caused by archaeological remains, would be also an interesting application.

Coordination of automated terrain vehicles, or surveillance are other two entries of a potentially long list

Thesis organisation

Each one of the challenges involved in the project requires a series of accurate investigations. The present work represent an initial step in the design of the system which will take care of the formation flight. In particular the possibility of the use of a vision system to perform the flocking is investigated.

Being the idea of a vision based aircraft flock directly inspired to real birds, seems pretty much obvious that the idea should work. However is not so obvious up to what extent the behaviour of a real bird can be emulated, using an aircraft and a vision system. Moreover is unknown how to realize this type of system.

A large number of the investigations involved in the design process will be based on computer simulations; throughout simulation will then also tested the designed systems. Is therefore extremely important the development of a correct simulation environment.

The first aim of this work is to *realize a software platform* suitable for simulation of vision based flocking algorithms. The second aim is to *analyse the effect of the limited vision* throughout simulations conducted on a concrete flocking algorithm.

A top down approach is chosen in the present work:

- In the first chapter is designed a full dynamic mathematical model of the aircraft.

  The not linear model is inspired to the wot4 aircraft; at low angle of attack the aircraft behaviour is correctly reproduced, while simplifications are introduced for not common flight conditions. To simulate the aircraft dynamic behaviour, a set of 12 differential equations is obtained analysing the physics involved in the flight. A step by step Runge Kutta method of integration is presented; this method is used in the second and third chapter to perform time history simulations. A numerical linearization of the model around predefined steady state point is then operated. The obtained linear model is then used to describe the aircraft characteristic natural modes.

- In the second chapter the control system for the designed aircraft model is developed.

  To stabilize the natural modes of the aircraft and to perform the desired manoeuvres three different autopilot are designed; a pitch hold autopilot, a speed hold autopilot and a jaw hold autopilot. All the three systems were designed on the linearized aircraft model, applying the classic control theory.

  All the software routines created in the two chapters are coded up as a single C library.

- In the last chapter a flocking algorithm is presented, and a series of investigation on the vision system are conducted.

  The flocking algorithm is based on the two rules of aggregation and avoidance as indicated by C. Reynolds. Simulations to demonstrate the aggregation capability are then shown. The dependence of the flocking capabilities from the max vision distance and from the view angle is investigated. A particular attention is also dedicated to the role of the aircraft cruise speed and of its maximum banking angle.

Chapter 2

Aircraft model

2.1 Introduction

As natural starting point to simulate a flock of air vehicles, in this initial chapter the aircraft mathematical is developed.

The specific aircraft model created in this chapter is for simplicity referred in all the work as Grid Swarm Aircraft Model (GSAM).

After a definition of the coordinate systems involved (§2.2), the force and moment equations are obtained (§2.3) and the not linear model is presented (§2.4). In the second part of the chapter, the simulation method is presented (§2.5); under the steady state flight assumption (§2.6), the GSAM is then linearized and its state space model is obtained (§2.7). In the latter paragraph (§2.8) the dynamic behaviour of the aircraft is deduced and analyzed .

2.2 Frames and coordinate systems

All this work is developed under the flat earth assumption; this assumption considers the earth as an infinite plane and so, centripetal and Coriolis acceleration acting on the aircraft are neglected. Due to the short range and slow speed flight capabilities of the GSAM, the effect of the centripetal and Coriolis acceleration is small. The flat earth assumption is therefore made in all this work. This assumption considers the earth as an infinite plane and so, centripetal and Coriolis acceleration acting on the aircraft are neglected. Under this assumption the "world" into which the aircraft fly could be defined by a right-handed coordinate system F_E (fig.2.1). In this work F_E is referred as the Euler frame.

The deformations of the aircraft structure during the flight are beyond the scope of this work, thus they are completely neglected. The aircraft is considered a rigid body and a second coordinate system F_B (fig.2.1) centred in the aircraft *c.g.* is introduced and denominated body frame. The exact position and orientation of the aircraft is specified by the *c.g.* coordinates and by the angular displacements (θ, ϕ, ψ) between the Euler and the body axes directions.

Several ways to perform the rotation from F_E to F_B are possible, so the notation (θ, ϕ, ψ) is ambiguous if is not defined how the displacement are specified. The most common convention in the aerospace field is adopted here. The triplet (θ, ϕ, ψ) identifies a sequence of three rotation defined as follow:

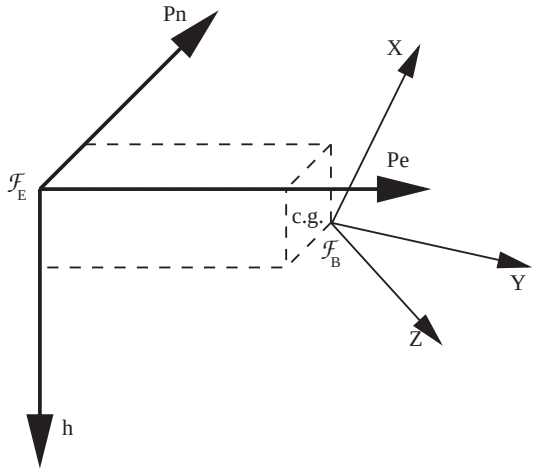

Figure 2.1: Euler frame F_E, body frame F_B

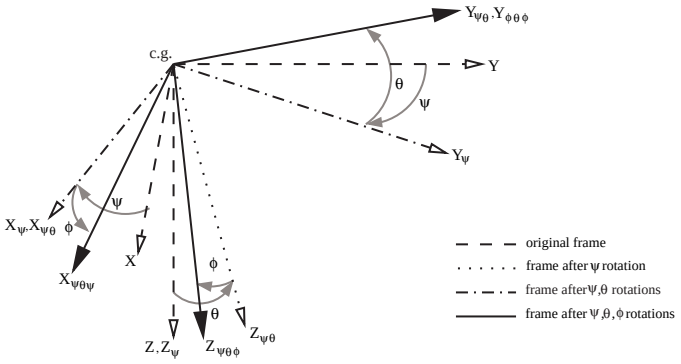

Figure 2.2: angular displacements (θ, ϕ, ψ) between the Euler and the body axes

1. right-handed rotation about the Z axis (positive ψ)
2. right-handed rotation about the new Y axis (positive θ)
3. right-handed rotation about the new X axis (positive ϕ)

where the sign of the angular displacement is defined according to the right hand rule. The angular rotation about the Y axis is commonly called *pitch*, *jaw* is named the rotation about the Z axis and the rotation about the X axis is denominated *roll*.

In the following paragraphs the derivative $\dot{\psi}$,$\dot{\theta}$,$\dot{\phi}$ and the angular velocity about the body axes P,Q,R are used, their positive direction is fixed in accordance with the right hand rule. The forward (U), lateral (W) and downward (V) speed in the body frame are defined with the same positive direction of the respective axis (fig.2.3).

In the body reference frame the X axis is aligned with the fuselage reference line; this make F_B suitable to compute the kinematic rigid body equation, but leads complications into the equations of the aerodynamic forces. All the aerodynamic forces are naturally referred to the true air flow direction, so is useful to introduce other two coordinate systems.

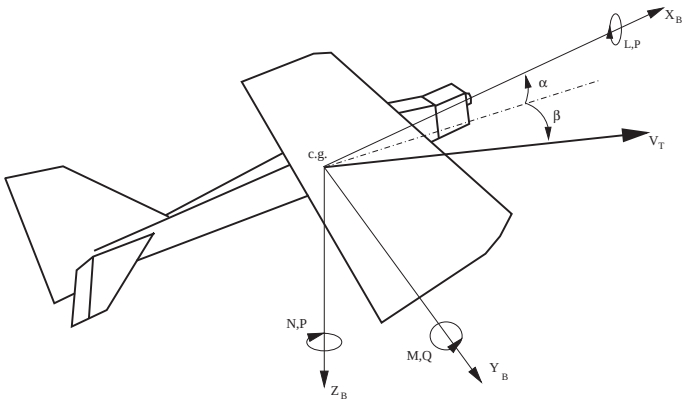

Figure 2.3: velocities, moments and angles sign convention

The first is the stability frame (F_S), it is defined as a planar rotation of F_B about the X_B axis with angular displacement α ($angleofattack$). This reference frame is important because all the moment equation are defined here.

The second is the wind reference system (F_W); it is obtained from the stability axis through a rotation about the Z_S axis with angular displacement β ($sideslip\ angle$). All the aerodynamic forces ($drag, lift, sideforce$) are specified in this frame. The X_W axis direction is the effective direction of the aircraft *c.g.*, and so it will determine the flight path of the aircraft.

In level flight and also during a steady state coordinated turn, the sideslip angle is zero, therefore F_W and F_S are coincident.

All the mathematical relation between the four coordinate system introduced so far (F_E, F_B, F_S, F_W) are reported in the appendix A.2.

State Vector

It is already clear that the static position of the aircraft is completely identified by the coordinate of its *c.g.* (P_e,P_n,h) and by its orientation respect to the Euler frame (ϕ,θ,ψ). These six variables cannot completely define the model because they do not represent its dynamic state. For this purpose to the six variables already chosen must be added the respective derivatives. The obtained set

$$\mathbf{X} = [\dot{P}_e, \dot{P}_n, \dot{h}, \dot{\phi}, \dot{\theta}, \dot{\psi}, P_e, P_n, h, \phi, \theta, \psi]$$

specifies completely the state of the model and so it is called state vector. Due to the fact that all the aerodynamic coefficients are defined as function of α,β, V_t,Q,R,P a different state vector was chosen for convenience of computation

$$\mathbf{X} = [V_t, \alpha, \beta, \theta, \phi, \psi, Q, P, R, P_n, P_e, h].$$

The two formulations are obviously equivalent because a mathematical relation (2.15 -2.17) is present between V_t,α,β,P,Q,R and $\dot{P}_e$, $\dot{P}_n$,$\dot{h}$,$\dot{\theta}$,$\dot{\phi}$,$\dot{\psi}$.

For simplicity also the inputs are collected in the input vector

$$\mathbf{U} = [\delta_e, \delta_a, \delta_r, \delta_t].$$

2.3 Forces and moments

In order to have a realistic model of the aircraft, the real physical forces acting upon it must be considered.

Assumption

During the analysis of the physical phenomena involved in the aircraft flight, will be also presented some simplifying assumption made. All this assumptions are based un considerations about the simulation environment that is developed in this work.

The first consideration is about the cruise altitude; due to the flying capabilities of the GSAM model and the typical application of a gridswarm is reasonable to think that the maximum altitude is only some hundreds of meters. This limitation allows considering constant all the parameters dependent from altitude or altitude variation.

The maximum speed capability of this type of aircraft is also limited, so all the effect depending in high Mach number can be obviously neglected.

Take off and landing operations will not be considered; only low angle of attack configuration are thus considered

The normal cruise condition for the entire flock is a constant speed level flight. The velocity matching is one of the requirement to have a flocking behaviour, thus the airspeed of a single aircraft will vary only near the cruise one. For the sake of simplicity all the dependence of the aerodynamic coefficients from airspeed are therefore neglected.

In the next chapter the control system is designed assuming that all the flying manouvre are suggested by a camera mounted on the aircraft. Due to the quite narrow view angle and reduced view distance of this camera, is shown in section 3.6 that the effective input command for the aircraft can seldom lead to a stall condition. Therefore the stall condition is only roughly modelled.

General formulation

For the sake of uniformity all the forces and moments are expressed as a product of the dynamic pressure ($\bar{q}$), an aerodynamic coefficients C_x (the subscript indicates the force or moment to which it is referred) and a typical area A

$$F_x = \bar{q} C_x A. \tag{2.1}$$

The dynamic pressure is expressed by

$$\bar{q} = \frac{1}{2}\rho V_t^2$$

where ρ is the air density and V_t is the airspeed of the free stream in which the aircraft is immersed. As stated by [12] using a model of the atmosphere the air density can be calculated from the altitude above the sea level and the air temperature at sea level. However in the present work the changes of the air density

are considered negligible [1] and so ρ is a constant equal to the sea level air density $\rho_{ssl} = 1.225\frac{kg}{m^3}$. Under this assumption the aircraft behaviour is completely independent from the aircraft absolute position, i.e. the same trimming configuration is correct at different altitude. Also the development of the control system is simplified due to the absence of the altitude contribution.

Aerodynamic coefficients

The dimensionless aerodynamic coefficient C_x in equation 2.1 represents the ability of the aircraft to produce a specific force or moment, i.e. the ability of the airfoil to produce drag. An aerodynamic coefficient is usually a complex function of the free air stream characteristics and of the aircraft state. The free air stream contribution is accounted through Reynolds number Mach number, the aircraft state contribution is accounted through the state variables and the control surface deflections.

The dependence of the aerodynamic coefficients from these terms is complex and not linear, so they are commonly implemented in a computer program in form of look-up tables. Complete table of the aerodynamic coefficients are usually derived from wind tunnel test or flight test measurement.

When this type of datas are not avaliable, especially if the aim of the model is to reproduce the aircraft in a restricted flight envelope (i.e. small angle of attack, low Mach number), the dependences can be made explicit.

The aerodynamic coefficients can be approximated as a sum of constant coefficients multiplied by state variables and input variables. This approach is used in the GSAM; the complete table of all the aerodynamic coefficients is reported on appendix A.1.

2.3.1 Lift

The lift force is conventionally defined as the component of the aerodynamical force orthogonal to the free stream velocity vector. This force can be written as

$$L = \bar{q}\bar{s}C_L \tag{2.2}$$

where C_L is the aerodynamic lift coefficient. C_L accounts the contribution of wings, fuselage and tail to the lift force. The lift coefficient is mainly dependent on α, and its dependence is linear below the stall point. The slope of the lift curve is determined by the aspect ratio and sweep-angle of the wings. At the stall point the lift coefficient drop sharply as α increase.

[1] According to [12] below 11000 geopotential meters the air density at a generic altitude (h) can be expressed by

$$\rho(h) = \rho_{ssl}(1 - 6.875585610^{-6}h)^{4.255863}$$

where ρ_{ssl} is the standard sea level air density ($1.225\frac{kg}{m^3}$).

Assuming for example a maximum altitude $h_{max} = 300m$ the fractional difference between the real air density and the air density at sea level is only

$$\frac{\rho(h_{max})}{\rho_{ssl}} = 0.003$$

thus negligible.

In the *GSAM* model, C_L was specified by

$$\begin{cases} C_L = C_{L_0} + C_{L_\alpha}\alpha + C_{L_{tail}}\delta_e & \alpha < \alpha_{ML} \\ C_L = C_{L_0} + C_{L_\alpha}\alpha_{ML} + C_{L_{tail}}\delta_e & \alpha \geq \alpha_{ML} \end{cases}.$$

Clearly from fig. 2.4 can be seen that the lift coefficient is fairly approximated for

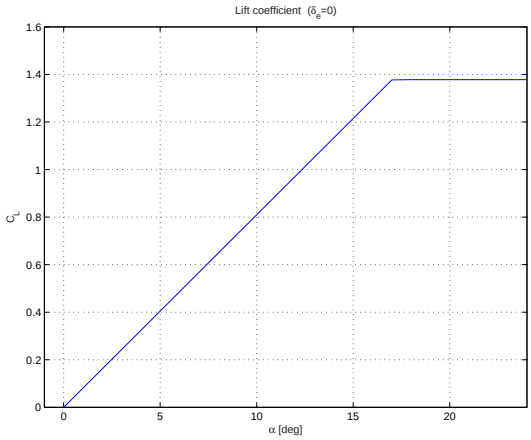

Figure 2.4: C_L vs α

low angle of attack, but the stall conditions is roughly simulated simply clamping the lift coefficient for $\alpha \geq \alpha_{ML}$.

The resulting model behaviour will not be closer to the reality but in force of the assumptions made this is acceptable. A more sophisticated model should also consider the effect of the dynamic pressure change on wings and tail caused by the propeller.

2.3.2 Drag

The drag force is defined as the component of the aerodynamic force in the direction of the free air stream velocity vector. It is a combination of friction drag and the drag generated when the integral of the pressure on the whole surface of the aircraft is not zero. How the the drag force is divided into the two components can strongly vary in relation to the flight condition. Drag can be expressed in the general form

$$D = \bar{q}\bar{s}C_D \tag{2.3}$$

where $\bar{s}$ is the wings reference area and C_D is the aerodynamical drag coefficient.

Below the stall conditions C_D is essentially proportional to the square of the lift coefficient (C_L). Beyond stall the expression of C_D is more complicated; there is still a parabolic dependence from C_L but it also depends on the angle of attack.

For the model adopted in this work, the coefficients of drag is supposed to be independent from the airspeed (as already explained) and simply proportional to the square of the lift coefficient (C_L) through the $C_{D_{CL}}$ coefficient

$$C_D = C_{L_0}C_{D_{CL}}C_L.$$

The coefficient C_{D_0} is needed because the minimum drag occurs at a non-zero value of C_L. The plot of C_D vs C_L is visible in fig. 2.5. The same consideration

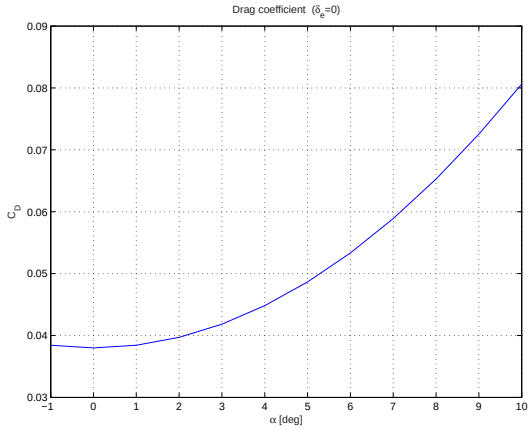

Figure 2.5: C_D vs C_L

about the stall expressed for C_L are also valid here.

In a more complex model also the additional drag contribute of landing gear or control surface could be accounted.

2.3.3 Sideforce

In a symmetric aircraft the sideforce is created essentially by sideslipping motion. The angular difference between the fuselage reference line and the free stream velocity vector lead to a lateral impact of air on the aircraft and so to a lateral force.

The sideforce can be expressed as

$$S = \bar{q}\bar{s}C_S \tag{2.4}$$

where C_S is the sideforce aerodynamic coefficient. C_S is mainly dependent on α and β; for low angle of attack the dependence from β is almost linear. The caracteristic of C_S is symmetrical respect to $\beta = 0$ and following the sign convention introduced in the previous section, a positive sideslip lead to a negative sideforce and vice versa.

In the *GSAM* C_S is expressed as a linear function of the rudder deflection and of the sideslip angle

$$C_S = C_{S_{\delta_r}}\delta_r + C_\beta\beta.$$

With $\beta = 0$ and the rudder in the default position the sideforce is thus zero.

2.3.4 Rolling moment

The rolling moment is principally a function of sideslip angle and aileron deflection. Following the by now usual formulation

$$l_s = \bar{q}\bar{s}\bar{c}C_l \tag{2.5}$$

where C_l is the rolling moment aerodynamic coefficient.

At low Mach number and small sideslip, the rolling coefficient is linear with β, but changes in α have a strong effect on the slope of C_l. As it easy to understand,

the value α determines the direction between the free stream velocity vector and the fuselage reference line, and thus the magnitude of the roll moment. The magnitude of the coefficients in the expression of C_l are closely related to the aircraft characteristic; in particular to the dihedral angle of the wings, and to the wings sweep.

In the *GSAM* the C_l coefficient is modelled by the equation

$$C_l = C_{l_\beta}\beta + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r + \frac{\bar{b}}{2V_t}(C_{l_P}P_{stab} + C_{l_R}R_{stab})$$

where

$$C_{l_\beta} = C_{l_{\beta_0}} + C_{l_{\beta_{C_L}}}C_L \qquad C_{l_R} = C_{l_{R0}} + C_{l_{R_{C_L}}}C_L.$$

The contributes of the control surfaces (rudder and ailerons) deflections to C_l are considered linear. The coefficient of the linear contribution of β to C_l is a function of the lift coefficient as can be better seen in fig.2.6. The resisting action to the

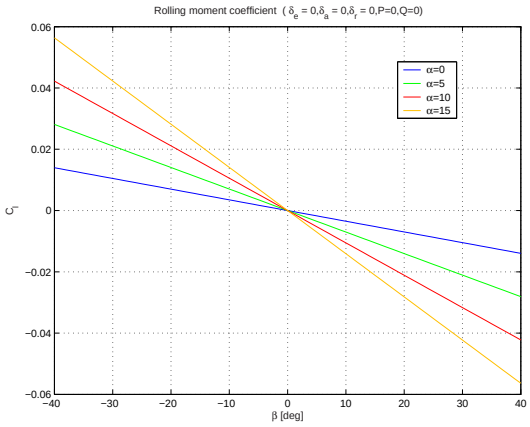

Figure 2.6: C_l dependence from α and β

rolling and yawing motion is also considered through the damping coefficients C_{l_P} and C_{l_R}. As it is for C_{l_β} also the proportional dependence of C_l from R is considered variable with α because both the contributions are generated by the same phenomenon.

2.3.5 Pitching Moment

For a low speed aircraft the pitching moment can be written as a function of α and of the elevator deflection. The pitch derivative and the α derivative are also introduced to model the resisting action to the change in pitch. The coefficients $C_{m_{\dot{\alpha}}}$ and C_{m_Q} are for this reason called damping coefficients.

The total pitching moment referred to the stability axes is defined as

$$m_s = \bar{q}C_m \tag{2.6}$$

with C_m defined as

$$C_m = C_{m_0} + C_{m_\alpha}\alpha + C_{m_{\delta_e}} + \frac{\bar{c}}{2V_t}C_{m_Q} + C_{m_{\dot{\alpha}}}\dot{\alpha}.$$

All the dependences involved are linear; the presence of the C_{m_0} coefficient will require a positive angle of attack to have a steady state flight condition.

In the *GSAM* model are not accounted additional effects induced by the engine. In a propeller aircraft in fact can be modelled the effect of the propeller which at low angle of attack yields the effective angle of attack independent from the airflow direction.

2.3.6 Yawing Moment

Yawing moment are create by sideslip and by the action of the rudder. At low sideslip angle, the yawing moment is quite linear with β, at high α the fuselage yawing moment and drop of dynamic pressure on the tail, can lead to a loss of directional stability. In the *GSAM* the yawing moment (referred to the stability axes) is defined by

$$n_s = \bar{q}\bar{s}\bar{c}C_n \tag{2.7}$$

where

$$C_n = C_{n_\beta} + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r + \frac{b^2}{2\bar{c}V_t}$$

$$C_{n_R} = C_{n_{R0}} + C_{n_{R_{C_L}}} C_L^2$$

The damping coefficient C_{n_R} is dependent on C_L^2 to model the intrinsic instability present at high angle of attack. The relation between α,β and C_n is also shown in graphical form on figure 2.7.

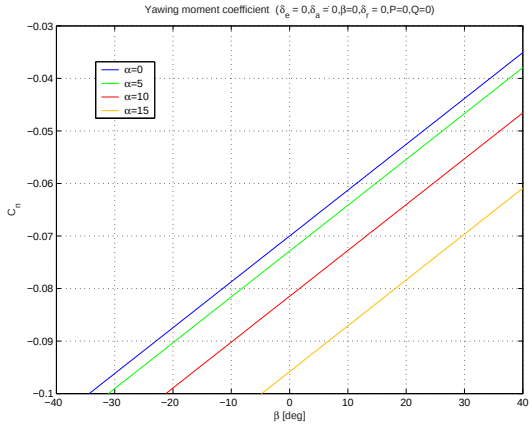

Figure 2.7: Dependence of C_n from α,β

A more deep analysis would also reveal other effects which can also contribute to the yawing moment. These are the propeller effect, unbalanced trust (in presence of two engines) and also the difference in lift between wings which rise when the aircraft is banked.

2.3.7 Thrust

Following a basic idea of simplicity for the model, the engine is modelled simply as a thrust force applied in the *c.g.* of the aircraft and aligned with the fuselage

reference line this assumption suppresses the contribution to the pitching moment given by changes of thrust.

However a maximum limit to the thrust force (6.26 N) was fixed to simulate the finite amount of engine power usually available in an aircraft.

2.4 Nonlinear aircraft model

We are now in the position to write all the equation of the nonlinear aircraft model considered like a rigid body. With the aim of a light illustration, a heavy use of matrix calculation is made in this section; all the matrices used are reported in detail on appendix A.2.

Force equations

The forces acting on the aircraft *c.g.* are the result of the vector sum of three different force vectors; these are the aerodynamical forces vector, the gravity forces vector and the thrust vector. While the thrust vector is referred to the body axes, the aerodynamical and gravity vectors are not; they must be then premultiplied respectively for the matrices $\mathcal{C}_{WB}$ [2] and $\mathcal{C}_{EB}$. The resulting expression is

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} T \\ 0 \\ 0 \end{bmatrix} - \mathcal{C}_{BW} \begin{bmatrix} D \\ S \\ L \end{bmatrix} + \mathcal{C}_{BE} \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{2.8}$$

where D, S, L are obtained from equations 2.2 - 2.4, T is the thrust force and mg is obviously the weight force. To obtain the acceleration of the *c.g.* in the three direction of F_b, the acceleration contribute coming from the rotation about the body axes must also be considered.

$$\begin{bmatrix} \dot{U} \\ \dot{V} \\ \dot{W} \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \frac{1}{m} - \begin{bmatrix} P \\ Q \\ R \end{bmatrix} \times \begin{bmatrix} U \\ V \\ W \end{bmatrix}. \tag{2.9}$$

Kinematic equations

The angular velocity about the F_E axes can be easily computed as a sum of P, Q and R decomposed respect to the Euler axes.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \mathcal{C}^*_{EB} \begin{bmatrix} P \\ Q \\ R \end{bmatrix}. \tag{2.10}$$

Moment equations

The moment's equations 2.5 - 2.7 are expressed in the stability axes coordinate system, so first the angular speeds referred to F_S must be computed.

[2] The rotation matrix $\mathcal{C}_{WB}$ indicates the transformation from the reference system F_E to the system F_W

$$\begin{bmatrix} P_{stab} \\ Q_{stab} \\ R_{stab} \end{bmatrix} = \mathcal{C}^*_{SB} \begin{bmatrix} P \\ Q \\ R \end{bmatrix}. \tag{2.11}$$

This expression of the speed can now be used in the equations 2.5 - 2.7 to compute l_s,m_s,n_s. To obtain the espression of the moments in the body axes another change of coordinates ($\mathcal{C}_{BS}$) is needed

$$\begin{bmatrix} l \\ m \\ n \end{bmatrix} = \mathcal{C}_{BS} \begin{bmatrix} l_S \\ m_S \\ n_S \end{bmatrix} + \mathcal{C}_{d_{c.g.}} \begin{bmatrix} L \\ S \\ D \end{bmatrix}. \tag{2.12}$$

The second term in the right side of the equation provides the contribution to the pitching moment due to the fact that the aerodynamical centre differ from the *c.g.*

The derivative of the angular velocity con now finally be calculated

$$\begin{bmatrix} \dot{P} \\ \dot{Q} \\ \dot{R} \end{bmatrix} = \mathcal{I}^{-1} \left(\begin{bmatrix} l \\ m \\ n \end{bmatrix} + \mathcal{C}_w \mathcal{I} \begin{bmatrix} P \\ Q \\ R \end{bmatrix} \right). \tag{2.13}$$

Navigation equations

The derivative of the *c.g.* position referred to F_E are obtained from the body axes speed (U,V,W) using the rotation matrix $\mathcal{C}_{EB}$

$$\begin{bmatrix} \dot{P}_w \\ \dot{P}_e \\ \dot{h} \end{bmatrix} = \mathcal{C}_{EB} \begin{bmatrix} U \\ V \\ W \end{bmatrix}. \tag{2.14}$$

Additional equations

This set of 12 ODE (equations 2.9,2.10,2.13, 2.14) presented so far allow to compute the derivative of the state vector. As already explained also α,β,V_t are used as state variables, therefore the equation to compute $\dot{\alpha}$,$\dot{\beta}$,$\dot{V}_t$ from equation 2.9 are also presented

$$\dot{\alpha} = \frac{U\dot{W} - W\dot{U}}{U^2 + W^2} \tag{2.15}$$

$$\dot{\beta} = \frac{\dot{V}V_t - V\dot{V}_t}{V_t\sqrt{U^2 + W^2}} \tag{2.16}$$

$$\dot{V}_t = \frac{U\dot{U} + V\dot{V} + W\dot{W}}{V_t}. \tag{2.17}$$

The whole set of differential equation, is referred in the following sections as a single function of the input vector $\mathbf{U}(t)$ and the state vector $\mathbf{X}(t)$

$$\dot{\mathbf{X}}(t) = f(\mathbf{X}(t), \mathbf{U}(t)). \tag{2.18}$$

2.5 Numeric aircraft model

In the previous paragraph a set of 12 ODE was derived to describe the physical phenomena acting upon the aircraft. Now a software program to implement that model is presented.

The differential equations derived are not linear, and the input signals are arbitrary, so an analytical solution of the equations is simply unthinkable; a numerical solution i required.

All the equations 2.8-2.14 were codified into a MATLAB function (reported in B.1) to easily compute the state derivatives ($\dot{\mathbf{X}}$) from the state ($\mathbf{X}$) and the input ($\mathbf{U}$) vectors

```
xdot=GSAM(x,u);.
```

Every state variable represent a physical energy stored in the system, so instantaneous change of the state variables are not allowed because this would require infinite power. All the state functions are therefore continuous and Lipschitzians (in a defined time interval $[t_0, t_1]$). This are conditions necessary and sufficient [13] to guarantee that the initial value problem

$$\begin{cases} \dot{\mathbf{X}}(t) = f(\mathbf{X}(t), \mathbf{U}(t)) \quad t \in [t_0, t_1] \\ \mathbf{X}(t_0) = \eta \end{cases} \tag{2.19}$$

has exactly one solution $\mathbf{X}(t)$ continuous and differentiable.

To evaluate numerically the aircraft trajectory, for a given input vector $\mathbf{U}(t_0)$ and an initial condition η, a discrete time sequence of state vector values

$$\mathbf{X}(t_0 + kT) \quad k = 1, 2, 3, 4, ...$$

must be computed. This idea of solution raises suddenly the problem of choosing the correct time step T and the correct method of integration.

Two basic types of integration methods are possible, the *fixed step* method and the *linear multistep method* (LMM).

Fixed step methods

The idea at the base of all the fixed step (or Runge Kutta) methods for a generic function $x(t)$ comes directly from the Taylor's series expansion. $x(t)$ can be written as

$$x(t_0 + T) = x(t_0) + T\dot{x}(t_0) + \frac{T^2}{2!}\ddot{x}(t_0) + \dots$$

if the terms of order higher then the first are neglected, x_E can simply be approximated as

$$x_E(t_0 + T) \approx x(t_0) + Tf(x(t_0), t_0).$$

This first order method is called Euler integration.

Using a weighted sum of the function $\dot{x}(t)$ calculated in different points of the time step, higher order methods are possible (conventionally a method is called of nth order if its error term is $O(T^{n+1})$). All this type of methods are explicit because only the input and the initial condition are needed to compute the next state of the system.

In one hand the implementation of this method is quite simple but in the other hand they can require a high volume of computation to achieve the required accuracy.

LMM

In the LMM the solution is a linear combination of past values of the function and current and past values of its derivative

$$x(n+1) = \sum_{r=0}^{n} a_r x(n-r) + T \sum_{r=-1}^{n} b_r \dot{x}(n-r). \tag{2.20}$$

Using this approach with a correct set of parameters a_i,b_i methods in which only one evaluation of $\dot{x}(t)$ per each time step is necessary can be designed. This types of methods tends thus to be computationally more efficient then the RK methods.

From 2.20 can be seen that these algorithms are not explicit because $x(n+1)$ is required to evaluate $\dot{x}(n+1)$. An initialization through a fixed step algorithm is then required for this type of methods.

More problematic then this will be the effect of a change in the equations of motion (i.e. due to limitation or saturation) during the simulation. In such case with the LMM the function estimation will be error affected because the previous values used in the computation were calculated using the "old" equations. Vice versa this would not be a problem for the fixed step methods.

Derivative limitation and controls saturation effects are introduced, in the last section of the next chapter during the design of the control system therefore this consideration and the simplicity of implementation, lead to prefer a fixed step method.

Methods comparison

To choose the most suitable fixed step method, a trade off between the accuracy required and the computational complexity must be made. The embedded Runge-Kutta formulas invented by Fehlberg can be used to estimate the error as proposed in [14]. The formulation of a RK method, (a fifth order in this case) can be written as:

$$\begin{aligned} k_1 &= Tf(x_n, y_n) \\ k_2 &= Tf(x_n + a_2 T, y_n + b_{21} k_1) \\ &\vdots \\ k_6 &= Tf(x_n + a_3 T, y_n + b_{31} k_1 + b_{32} k_2 + \ldots + c_6 k_6) \end{aligned} \tag{2.21}$$

$$y_{n+1} = y_n + c_{51} k_1 + c_{52} k_2 + c_{53} k_3 + c_{54} k_4 + c_{55} k_5 + c_{56} k_6 + O(T^6) \tag{2.22}$$

$$y^*_{n+1} = y_n + c_{41} k_1 + c_{42} k_2 + c_{43} k_3 + c_{44} k_4 + c_{45} k_5 + O(T^5) \tag{2.23}$$

and the error estimate is

$$\Delta = y_{n+1} - y^*_{n+1}. \tag{2.24}$$

The coefficients k_i,b_{ij},c_{ij} are determined to have an efficient method and good error estimation. Lots of possibilities to choose these coefficient can be found in the literature; here a $CSIRK$ set (table 2.1) proposed by Verner [15] is used

i	a_i	b_{ij}				
1	0					
2	$\frac{1}{4}$	$\frac{1}{4}$				
3	$\frac{4}{13}$	$\frac{5}{169}$	$\frac{50}{169}$			
4	$\frac{65}{68}$	$\frac{323375}{314432}$	$\frac{-283075}{78608}$	$\frac{1109485}{314432}$		
5	1	$\frac{11729}{7475}$	$\frac{-130}{23}$	$\frac{299468}{58075}$	$\frac{-55488}{754975}$	
6	$\frac{1}{2}$	$\frac{-819}{4600}$	$\frac{65}{46}$	$\frac{-3225027}{3716800}$	$\frac{35836}{58075}$	$\frac{-31}{64}$
$j=$		1	2	3	4	5

i	c_{ij}					
1	$\frac{91}{750}$	0	$\frac{485537}{909000}$	$\frac{2672672}{3522375}$	$\frac{-161}{360}$	$\frac{46}{1395}$
2	$\frac{116}{975}$	0	$\frac{68107}{121200}$	$\frac{78608}{98475}$	$\frac{-23}{48}$	
3	$\frac{163}{1950}$	0	$\frac{9971}{15150}$	$\frac{25432}{98475}$		
4	$\begin{cases} \frac{-3}{10} \\ -1 \end{cases}$	$\begin{cases} 0 \\ 2 \end{cases}$	$\frac{13}{10}$			
5	1					
$j=$	1	2	3	4	5	6

Table 2.1: Verner parameters for embedded RK methods

because it provides the necessary coefficients for a fifth order method and also for all its lower order embedded methods.

From this coefficients using the equation 2.21-2.24, a software program for the 3^{rd},4^{rd} and 5^{th} order methods was codified (B) and a numerical comparison was made to choose the most suitable one.

With a set of 12 ODE, where the variables differ considerably in magnitude, a more appropriate way then equation 2.24 to deal with the error, is not to consider the absolute error but the fractional error

$$\epsilon = \frac{\Delta_0}{y}.$$

In the present situation a set of ODE is solved, so the fractional error in each equation is computed and the worse one is used to set the time step (T) of the algorithm.

A 200 s time history simulation using the $GSAM$ model was run; for each algorithm the time step was adjusted to obtain an error of fractional magnitude about 5%. The result obtained are presented in table 2.2.

time step [s]	**error** ϵ	**number of function evaluation**	**method order**
0.075	0.0496	13332	3
0.10	0.0556	12500	4
0.18	0.0652	8328	5
0.10	0.00413	15000	5

Table 2.2: RK 3^{rd},4^{rd} and 5^{th} order comparison

As expected, for a given accuracy, high order methods allow larger time step. Therefore even if the method itself requires more function evaluations, the reduced number of steps reduces the total number of function evaluations required. The 5^{th} order method could seem to be the most suitable choice, however the selection of the integration step is not only driven by accuracy constriction.

In fact the integration method takes the implicit assumption that the input are invariant during the time step so also this requirement must be fulfilled to have correct results.

In chapter 3 a numerical algorithm with a time step of 10 ms is derived to provide the necessary input for the GSAM; as consequence the 5^{th} order method with 18 ms time step cannot be used. One possibility is to use the 5^{th} order methods with a time step of 10 ms but as can be seen in the last row of the table 2.2, this choice gives accuracy higher then the requirements but will almost double the computation requirements. The 4^{th} order method becomes thus the better trade off.

This effort spent to choose the integration method, is necessary because in the flocking simulation several GSAM are used at the same time so a good integration method can guarantee a faster simulation.

2.6 Steady state flight

The steady-state aircraft flight is defined as the equilibrium condition in which all the forces and moment (referred to the body frame) are constant or zero. This condition is important because it can be used as an initial state for the time history simulation. Furthermore the behaviour of the system near to a singular point can be deduced examining slightly perturbation of the state variables around the equilibrium value. The linearization proposed on the next section is based on this principle, and so on the equilibrium point derived here.

Under the flat earth assumption, the forces and moments are not dependent on the aircraft *c.g.* position coordinates (P_e,P_n,h), therefore the steady-state condition does not impose constraint on these variables.

Different steady-state conditions can be obtained in relation to the constraint imposed to the other state variables:

- ***steady-state level flight***

$$\dot{P}, \dot{Q}, \dot{R}, \dot{V}_t, \dot{\alpha}, \dot{\beta}, \phi, \dot{\phi}, \dot{\theta}, \dot{\psi} \equiv 0, \text{ all control fixed} \tag{2.25}$$

- ***steady-state turn***

$$\dot{P}, \dot{Q}, \dot{R}, \dot{V}_t, \dot{\alpha}, \dot{\beta}, \dot{\phi}, \dot{\theta} \equiv 0, \text{ all control fixed} \quad \dot{\psi} = \text{turn rate} \tag{2.26}$$

- ***steady-state pull-up***[3]

$$\dot{P}, \dot{Q}, \dot{R}, \dot{V}_t, \dot{\alpha}, \dot{\beta}, \phi, \dot{\phi}, \dot{\psi} \equiv 0, \text{ all control fixed} \quad \dot{\theta} = \text{pull-up rate} \tag{2.27}$$

- ***steady-state roll***[4]

$$\dot{P}, \dot{Q}, \dot{R}, \dot{V}_t, \dot{\alpha}, \dot{\beta}, \dot{\theta}, \dot{\psi} \equiv 0, \text{ all control fixed} \quad \dot{\phi} = \text{roll rate} \tag{2.28}$$

[3] only instantaneous condition
[4] only instantaneous condition

Trim algorithm

The complex functional relation between the state variables makes practically impossible to calculate analytically a steady state condition. A minimization algorithm can be used to find the appropriate control inputs and independent state variable needed to achieve the desired equilibrium condition.

The idea is that the minimization algorithm (i.e. the simplex algorithm) will adjust some of the input and some of the state variables of the function

$$\dot{\mathbf{X}}(t) = f(\mathbf{X}(t), \mathbf{U}(t)) \tag{2.29}$$

in order to minimize a given cost function. The cost function must be expressed in terms of $\dot{\mathbf{X}}(t)$ so as its minimization leads also to the fulfilment of the steady-state constraint.

In the simulation developed in the last chapter, the steady-state level flight at constant speed (V_{t_e}) is considered as the normal cruise condition, therefore now the aircraft is trimmed in this situation.

The variables adjusted by the minimization program and the cost function must be decided in relation to the steady-state condition desired.

The condition 2.25 states that ϕ,P,Q,R must be zero, moreover V_t must be equal to the cruise speed V_{t_e}; therefore this variable are fixed to this value and cannot be changed by the algorithm. The condition of level flight impose $\theta = \alpha$, but does not set their value. These and all the remaining variables (β,$\mathbf{U}$) are adjusted by the trim algorithm.

Intuitively the cost function has to measure the "distance" of the state derivative vector from the steady state condition. A straightforward formulation for the cost function is then

$$C = \dot{V}_t^2 + \dot{\beta}^2 + \dot{\alpha}^2 + \dot{P}^2 + \dot{Q}^2 + \dot{R}^2;$$

if minimized, the condition 2.25 is verified.

The MATLAB implementation of the simplex algorithm was now used, and the results obtained for several different cruise speeds are reported in table 2.3. The equilibrium points are spread all over the flight envelope between the min-

$\mathbf{V_{t_e}}$ $[m/s]$	$\theta = \alpha$ $[rad]$	$\delta_\mathbf{e}$ $[rad]$	$\delta_\mathbf{a}$ $[rad]$	$\delta_\mathbf{r}$ $[rad]$	$\delta_\mathbf{t}$ $[N]$
11	0.1871	0.078	0	0	2.34
12	0.157	0.059	0	0	2.31
14	0.115	0.032	0	0	2.44
15	0.099	0.022	0	0	2.57
18.39	0.065	0	0	0	3.26
22	0.044	-0.013	0	0	4.32
25	0.033	-0.020	0	0	5.42
28	0.026	-0.025	0	0	6.69
30	0.022	-0.027	0	0	7.63
32	0.019	-0.029	0	0	8.64
33	0.018	-0.030	0	0	9.18

Table 2.3: Equilibrium points

imum and maximum steady state flight speed. The maxim speed was obtained using as input the maximum allowed thrust and trimming the aircraft for level flight. The minimum speed was identified as the speed permitted by an angle of attack just below the stall condition.

The simplex is an iterative algorithm, it stops the attempt to minimize the cost function when no further improvement are registered or when a maximum number of iteration is reached. An initial set of value for the variable to adjust (usually close to the expected result) must also be given as input. The complete MATLAB code used for the trimming procedure is reported in appendix B.3.

2.7 State space model

Since the aircraft model is not linear, before starting the design of a control system using the classical control theory (cap. 3), the model must be linearized. After a linearization conducted near an equilibrium point the model can be written in the space state form. From this formulation the dynamic behaviour of the aircraft can be analysed and the control system can be designed.

Under the restriction of wings-level non sideslip steady state flight is possible to perform an algebraic linearization.

However a more general way to approach the problem without introducing restriction, is to use a numerical linearization.

Linearization method

The state derivative can be estimated exploiting the response of the state equation

$$\dot{\mathbf{X}} = f(\mathbf{X}, \mathbf{U}). \tag{2.30}$$

to a small perturbation of the equilibrium condition. The multivariable Taylor's series expansion of the state equation can be written as

$$\dot{\mathbf{X}} - \delta\dot{\mathbf{X}} = f(\mathbf{X}_e, \mathbf{U}_e) + \delta\mathbf{X}\frac{\partial f}{\partial \mathbf{X}} + \delta\mathbf{U}\frac{\partial f}{\partial \mathbf{U}} + O(\delta\mathbf{X}^2, \delta\mathbf{U}^2). \tag{2.31}$$

If $(\mathbf{X}_e, \mathbf{U}_e)$ is an equilibrium condition obtained for example using the trim algorithm,

$$\dot{\mathbf{X}} = f(\mathbf{X}_e, \mathbf{U}_e) \equiv 0$$

therefore neglecting the high order terms in the equation 2.31

$$\delta\dot{\mathbf{X}} = \delta\mathbf{X}\frac{\partial f}{\partial \mathbf{X}} + \delta\mathbf{U}\frac{\partial f}{\partial \mathbf{U}}. \tag{2.32}$$

Which is the LTI formulation desired in the form

$$\dot{x} = \mathcal{A}x + \mathcal{B}u$$

where $\mathcal{A}, \mathcal{B}$ are commonly called Jacobian matrices. They represent the state derivative contribute due respectively to the state vector and to the input vector. Adding the equation

$$y = \mathcal{C}x + \mathcal{D}u$$

to the previous one the general state space formulation of the linear model is obtained. $\mathcal{C}$ and $\mathcal{D}$ are matrices used to compute the output vector y and the lowercase $\dot{x}$,x,u denote perturbation from the equilibrium point.

Using a forward backward approach each row of the Jacobian matrices can be easily calculated

$$\delta\dot{\mathbf{X}}_{fw} \equiv f(\mathbf{X}_e + [0,\ldots,\delta_{x_i},\ldots],\mathbf{U}_e) = \delta\dot{\mathbf{X}}_e + \begin{bmatrix} \frac{\partial x_1}{\partial x_i} \\ \frac{\partial x_2}{\partial x_i} \\ \vdots \end{bmatrix} \delta x_i + \begin{bmatrix} \frac{\partial x_1^2}{\partial^2 x_i} \\ \frac{\partial x_2^2}{\partial^2 x_i} \\ \vdots \end{bmatrix} \delta x_i^2 + \ldots \tag{2.33}$$

$$\delta\dot{\mathbf{X}}_{bw} \equiv f(\mathbf{X}_e - [0,\ldots,\delta_{x_i},\ldots],\mathbf{U}_e) = \delta\dot{\mathbf{X}}_e - \begin{bmatrix} \frac{\partial x_1}{\partial x_i} \\ \frac{\partial x_2}{\partial x_i} \\ \vdots \end{bmatrix} \delta x_i + \begin{bmatrix} \frac{\partial x_1^2}{\partial^2 x_i} \\ \frac{\partial x_2^2}{\partial^2 x_i} \\ \vdots \end{bmatrix} \delta x_i^2 + \ldots \tag{2.34}$$

$$\begin{bmatrix} \frac{\partial x_1}{\partial x_i} \\ \frac{\partial x_2}{\partial x_i} \\ \vdots \end{bmatrix} = \frac{\delta\dot{\mathbf{X}}_{fw} - \delta\dot{\mathbf{X}}_{bw}}{2\delta_{x_i}}. \tag{2.35}$$

The same procedure repeated for all the state variables and inputs gives the $\mathcal{A}$ and $\mathcal{B}$ matrices.

To code up this method in a computer program the magnitude of the perturbations must be defined. As it is clear this problem is related to the behaviour of the derivatives near to the equilibrium point. If the perturbation is too large the truncation error introduced neglecting the high order terms will be large; if conversely the perturbation is too small the round off error will affect the derivative estimation.

A straightforward solution is to decrease the magnitude of the perturbation until the derivative obtained converges [5] to a value. The initial value was chosen as a proportion of the derivative (10%) or a fixed value of 0.1 if the derivative was zero. In appendix B.4 the MATLAB function written is reported.

The obtained $\mathcal{A}$,$\mathcal{B}$ Jacobian matrices calculated on the second of the equilibrium points of table 2.3 are respectively

V_t	α	β	θ	ϕ	ψ	Q	P	R	P_e	P_n	h
−0.1538	3.8561	0	−9.8100	0	0	0	0	0	0	0	0
−0.0574	−8.2042	0	−0	0	0	1	0	0	0	0	0
0	0	−0.9108	0	0.5322	0	0	0.0650	−0.9979	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	1	0.0651	0	0	0
0	0	0	0	0	0	0	0	1.0021	0	0	0
0.2386	−87.184	0	0	0	0	−14.535	0	0	0	0	0
0	0	−6.8983	0	0	0	0	−3.2341	0.1188	0	0	0
0	0	17.173	0	0	0	0	−0.1683	−0.9971	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
0	0	18.395	0	−1.1949	18.395	0	0	0	0	0	0
−0	18.395	0	−18.395	0	0	0	0	0	0	0	0

[5] two consecutive iteration differ less then a predefined tolerance fixed to 1 10^{-6} in the present case

$$\begin{array}{c} \begin{matrix} \delta_e & \delta_a & \delta_r & \delta_t \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0.4339 \\ -0.7006 & 0 & 0 & -0.0015 \\ 0 & 0 & 0.3258 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -185.72 & 0 & 0 & 0.0064 \\ 0 & -6.2983 & 2.2681 & 0 \\ 0 & -8.1993 & -14.162 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}.$$

Decoupling

A careful examination of the Jacobian matrices reveal a clear decoupling between lateral and longitudinal motion. Note that the coefficient of the 3^{rd},5^{th},6^{th},8^{th} and 9^{th} row of the columns of V_T,α,θ,Q are zero and also are zero the coefficients of the 1^{st},2^{nd},4^{th} and 7^{th} row of the β,ϕ,ψ,P,R columns. To understand why, the force equations 2.9 and the moment equations 2.11 are here rewritten in a more suitable form and re-examined.

$$\begin{cases} \dot{U} = (R\sin\beta - Q\sin\alpha\cos\beta)V_t + (T - D\cos\alpha + L\sin\alpha)/m - g\sin\theta \\ \dot{V} = (P\sin\alpha - R\cos\alpha)V_t\cos\beta + S/m + g\sin\phi\cos\theta \\ \dot{W} = (Q\cos\alpha\cos\beta - P\sin\beta)V_t - (D\sin\alpha + L\cos\alpha)/m + g\cos\phi\cos\theta \end{cases} \tag{2.36}$$

It is quite easy to see that to have a decoupling of the second equation from the remaining two, ϕ must be zero and β must be small ($\therefore \sin\beta \simeq 0$).

The re-arranged kinematic equations

$$\begin{cases} \dot{\phi} = P + \tan\theta(Q\sin\phi + R\cos\phi) \\ \dot{\theta} = Q\cos\phi - R\sin\phi \\ \dot{\psi} = (Q\sin\phi + R\cos\phi)/\cos\theta \end{cases} \tag{2.37}$$

also show that the condition $\phi = 0$ lead to a complete independence of the second equation of the set if P,Q have small value.

Finally the imposition of this conditions on the moment equations

$$\begin{cases} \dot{P} = (I_{XZ}(I_X - I_Y + I_Z)PQ - (I_Z(I_Z - I_Y) + I_{XZ}^2)QR + I_Z l + I_{XZ} n)/\Gamma \\ \dot{Q} = (I_Z - I_X)PR - I_{XZ}(P^2 - R^2) + m \\ \dot{R} = ((I_X(I_X - I_Y) + I_{XZ}^2)PQ - I_{XZ}(I_X - I_Y + I_Z)QR + I_{XZ} l + I_Z n)/\Gamma \end{cases}$$

$$\Gamma = (I_X I_Z - I_{XZ}^2). \tag{2.38}$$

confirms the independence of the pitching moment from the yawing and rolling ones, thus a complete decoupling.

The three conditions which validate the decoupling are therefore

$$\phi = 0, \qquad \beta, P, R \; small.$$

The equilibrium point represented by a steady-state level flight, fulfil all these requirements and so it leads to a decoupled matrix. Both, the analysis of the

dynamical behaviour of the system and the control system are developed under the decoupling simplification.

This assumption is a common way to deal with the complexity of the aircraft model; it has a reasonable importance because most of the flying time is spent by the aircraft in this configuration. However validation and extension of the result obtained exploiting the decoupling must be provided when it is not guaranteed.

2.8 Dynamic behaviour

Exploiting the decoupling introduced jet, the longitudinal and lateral modes of the aircraft are now examined.

2.8.1 Longitudinal modes

Selecting the V_t,α,θ,Q columns from $\mathcal{A}$, the Jacobian matrix for the longitudinal dynamic was first of all obtained. Then using the MATLAB function `eig(A)` its eigenvalues an eigenvectors were computed.

$$
\begin{array}{l}
-0.0699 \pm j0.571 \quad (phughoid \quad \zeta = 0.1215 \quad \omega = 0.5750\ [rad/s] \quad \therefore \tau = 10.927s) \\
-11.3767 \pm j8.787 \quad (short-period \quad \zeta = 0.7914 \quad \omega = 14.3751\ [rad/s] \quad \therefore \tau = 0.437s)
\end{array}
$$

$$
\begin{array}{ccccccc}
\multicolumn{3}{c}{short-period} & & \multicolumn{3}{c}{phughoid} & \\
\begin{bmatrix} 0.0207 \\ 0.0360 \\ 0.0546 \\ -0.9916 \end{bmatrix} & \pm j & \begin{bmatrix} -0.0187 \\ -0.1000 \\ -0.0422 \\ 0 \end{bmatrix} & & \begin{bmatrix} 0.9977 \\ -0.0029 \\ -0.0097 \\ 0.0338 \end{bmatrix} & \pm j & \begin{bmatrix} 0 \\ -0 \\ 0.0580 \\ 0.0015 \end{bmatrix} & \begin{matrix} V_t \\ \alpha \\ \theta \\ Q \end{matrix}
\end{array}
$$

Two complex pair of eigenvalues are present, so the system is characterized by two oscillatory modes. The very different time constant makes very easy two identify the two modes as short-period mode and phughoid mode.

Short-period mode

In this flight condition the short-period mode is well damped. The larger contribute in the short-period eigenvector is given by the airspeed, while the remaining variables α,θ,Q seems to have quite the same contribute to this mode.

To have an insight about the relation between the aircraft parameters and this mode, will be used some algebraic expressions derived in [16]. In the assumption of decoupling, low Mach number and oscillatory short period mode, the following approximation of ω_{sp} and ζ_{sp} can be written

$$\omega_{sp}^2 \approx \frac{\bar{q}\bar{s}\bar{c}^2}{I_y}\left[-C_{m_\alpha} - \frac{\rho\bar{s}\bar{c}}{4m}C_{L_\alpha}C_{m_q}\right] \tag{2.39}$$

$$\zeta_{sp} \approx \frac{1}{4}\sqrt{\frac{\rho\bar{s}\bar{c}^3}{2I_y}}\frac{-C_{m_q} + \frac{2I_Y}{m\bar{c}^2}C_{L_\alpha} - C_{m_{\dot{\alpha}}}}{\sqrt{-C_{m_\alpha} - \frac{\rho\bar{s}\bar{c}}{4m}C_{m_q}C_{L_\alpha}}} \tag{2.40}$$

This two equation shows that the short-period mode intimately relates α and Q. An excitation of this mode (i.e. by an elevator doublet pulse) will lead to an oscillatory torsion about the Y body axes. The frequency of the oscillation and its damping are mainly proportional to the square root of the dynamic pressure and to the aerodynamic coefficients dependent from α and Q.

Phughoid mode

The phughoid eigenvectors shows that the main contribution to this mode is given by the airspeed and θ. This mode can be excited for example by a throttle doublet pulse, which will give as result an oscillation in airspeed and in θ In this flight situation the damping coefficient is quite low, but its long period make it easy to control. With the same assumption used for the short-period mode, w_p can be written as

$$\omega_{sp}^2 \approx \frac{2g}{\bar{c}} \frac{C_{m_\alpha} 2C_L}{C_{m_q} C_{L_\alpha} + \frac{4m}{\rho s \bar{c}} C_{m_\alpha}}. \tag{2.41}$$

From this relation appear clearly that the characteristics of the mode depend on the same aerodynamic coefficients on which the short period depends. Therefore as is shown in chapter 3 any action undertaken to change the phughoid dynamic will also effect the short-period mode.

2.8.2 Lateral and directional modes

From the complete Jacobian matrix of the system also the Jacobian matrix of the longitudinal dynamic was obtained (selecting the β,ϕ,P,Q columns), and its eigenvectors were calculated.

$$\begin{array}{rl} -0.8877 \pm j4.2292 & (\textit{dutch roll} \quad \zeta = 0.2054 \quad \omega = 4.3213\ [rad/s] \quad \therefore \tau = 1.454s) \\ -0.0031 & (\textit{roll subsidence} \qquad \tau = 103.09s) \\ -3.3619 & (\textit{spiral} \quad \tau = 0.2979s) \end{array}$$

$$\begin{array}{ccccc} \textit{Dutch Roll Mode} & & & \textit{Roll Mode} & \textit{Spiral Mode} \\ \begin{bmatrix} 0.0032 \\ -0.0287 \\ -0.2712 \\ 0.9181 \end{bmatrix} & \pm j & \begin{bmatrix} -0.2244 \\ -0.0560 \\ 0.1710i \\ 0 \end{bmatrix} & \begin{bmatrix} -0.0162 \\ 0.2842 \\ -0.9573 \\ 0.0496 \end{bmatrix} & \begin{bmatrix} -0.0255 \\ -0.8918 \\ 0.0380 \\ -0.4501 \end{bmatrix} \end{array} \quad \begin{array}{c} \beta \\ \phi \\ P \\ R \end{array}$$

Three modes are present in this case; an oscillatory mode (Dutch Roll Mode) and two stable exponential modes (roll subsidence mode and spiral mode).

Dutch Roll Mode

This mode involves all the longitudinal variables, it is thus basically a mixture of yawing rolling and sideslipping. The typical way to excite this mode is trough a rudder pulse, the resulting oscillation is quite fast ($\tau = 1.454s$) and light damped. If like in these flight conditions, the magnitude of C_{l_β} is small, the rolling motion could be neglected and an approximation of ω_d and ζ_d can be made [16].

$$\omega_d^2 \approx \frac{\bar{q}\bar{s}b}{I_Z}\left[C_{n_\beta} + \frac{\rho\bar{s}b}{4m}C_{Y_\beta}C_{n_r}\right] \tag{2.42}$$

$$\zeta_d \approx -\frac{1}{4}\sqrt{\frac{\rho\bar{s}\bar{b}^3}{2I_Z}}\frac{C_{n_r} + \frac{2I_Z}{mb^2}C_{Y_\beta}}{\sqrt{C_{n_\beta} + \frac{\rho\bar{s}b}{4m}C_{n_r}C_{Y_\beta}}}. \tag{2.43}$$

Both the frequency and the damping of the oscillation are proportional to the square root of the air density.

Roll subsidence mode

From the eigenvector of the Roll subsidence mode, is easy to see how this mode involves mainly the roll angle and the respective roll rate. The time constant of the exponential mode gives an idea of how much time the aircraft takes to start a roll manouvre. An approximated estimation of the time constant

$$\tau_r \approx \frac{1}{-\rho V_T \frac{\bar{s}b^2}{4I_X}C_{l_p}\left[1 - \frac{C_{l_\beta}C_{n_p}}{C_{n_\beta}C_{l_p}}\right] - \frac{gI_ZC_{l_\beta}}{V_TI_XC_{n_\beta}}} \tag{2.44}$$

shows that the time constant is inversely proportional to the product of air density and airspeed. As reasonable the roll response will be quicker at low speed and higher altitude.

Spiral mode

The last mode involves roll angle and yaw rate, its time constant represent the time needed to start a yaw manouvre. The much longer time constant respect the intuitive idea that the yawing manouvre is a result of the change of the lift force direction produced by a roll manouvre. The approximated expression of τ_s

$$\tau_s \approx \frac{\frac{V_T}{g}(C_{l_\beta}C_{n_p} - C_{n_\beta}C_{l_p}) - \frac{4I_Z}{\rho V_T b^2 \bar{s}}C_{l_\beta}}{(C_{l_\beta}C_{n_r} - C_{n_\beta}C_{l_r})} \tag{2.45}$$

confirms this closed relation between the two type of motion, the coefficients appearing in the 2.44 are in fact the same of the 2.45. In the next chapter is clearly shown that every change on one of these modes will affect the others.

Chapter 3

Control System

3.1 Introduction

In this chapter is designed a control system to stabilize and fly the GSAM.

In the last years, modern control theory have been intensively used for the design of aircraft control systems. The peculiarities of this approach are a design directly based on the state-space model and precise expressions for the estimation of the control system performance.

Despite these clear advantages of the modern control, the classic control is still used in the design of modern aircrafts.

Two are the motivation that support the classic control:

- is a well known and established method;
- gives more insight on the several concurrent effects present in an aircraft.

The GSAM is not a very complex model, but a good understanding of it is regarded as essential for the next stages of this work. Considering this, is more instructive to prefer the classic control system approach.

The aim of the control system is to be an intermediary between the flocking algorithm and the control inputs of the aircraft. The control system has to provide the aircraft model with the input vector necessary to obtain the manoeuvres suggested by the flocking algorithm, moreover it has to suppress all the undesired effects (i.e. oscillations) due to the aircraft dynamic behaviour.

After an important section about design criteria and requirements (§3.2), the design is divided in two parts; the longitudinal control system (§3.3) and the lateral control system (§3.4). The hard limits of the controller are accounted in §3.6, while the airspeed dependence is inspected in §3.5. Finally to validate the obtained controller a complete time history simulation is presented (§3.7).

3.2 Design criteria

The first step in the design of a control system is always to fix the set of controlled variables and a set of requirements.

The inputs of the control system are:

- V_{Td} the desired airspeed;

- ϕ_d the desired aircraft heading;
- θ_d the desired aircraft pitch;

while the controlled output variables are:

- δ_e elevator deflection;
- δ_a aileron deflection;
- δ_r rudder deflection;
- δ_T thrust setting.

A classic approach is followed the design is split in two different autopilots.

The longitudinal autopilot provides the pitch control (pitch auto hold, PAH) and the speed control (speed auto hold, SAH), while the lateral-directional autopilot provides the heading control (yaw auto hold, YAH).

Dealing with systems which behave like second order system, the design can be based on the three parameters time constant, damping coefficient and steady state error.To fix the appropriate requirements for these parameters a step back to the real scenario is needed to understand clearly their effects.

Time constant

The time necessary to complete a desired maneuvre of pitching or heading determines essentially two flock characteristics[1]:

- capacity of flock formation (higher manoeuvrability leads to reduce the number of situations in which the target cross the vision field of the aircraft but the aircraft is not quick enough to follow it);
- stability of the flock (fast control decrease the risk of collision);
- robustness of the flock (higher agility allows each boid to follow its neighbours also during fast direction change).

If not other constraints are involved, the control system is designed to have the shortest time constant with the aim of improving the flocking capabilities.

Damping

In the particular application developed in this work, the damping of the several oscillatory modes present assumes a great importance. Even if the vision algorithm is not implemented jet, the control system must be designed to simplify its realization. Independently from the algorithm used is well known that the complexity of vision tracking is strictly related to the quantity of motion that affects the target and the scene. An effort must be made to limit all the unnecessary motion, therefore oscillation, in particular high frequency oscillation, can not be tolerated.

[1] Confirmation to these consideration came also from observation made by ornithologists; typically a flock of small birds (higher manoeuvring capacity) is larger in number then a flock of big birds.

In the following sections the PAH and YAH systems are designed to have a deadbeat step response, whether not possible they are designed to have a good damping with a low natural frequency.

This last requirement is considered more important then the time constant requirement because its effect on the system can be more detrimental. The time constant requirement is thus fulfilled only if not in contrast with the damping one.

Steady state error

To properly design the control system for every controlled variable the maximum allowed steady state error is needed. To have a good cohesion of the flock, large steady state error can not be allowed for θ_d and ψ_d.

Is well known that the velocity matching is one of the most important capabilities asked to an actor of the flock. In fact if a boid is able to match the velocity of its flock mates it will probably not collide with them at list in the near future. The requirement for the speed steady state error has to be stricter; the steady state error is required to be null.

Autothrottle

The design of the SAH autopilot follows a slight different way.

From section 2.3.1 is well known that a change of the airspeed changes the lift force acting on the aircraft and therefore its altitude. If the pitch control is used to fly with a defined altitude (i.e. a target is being tracked) as result of the altitude change also the the desired pitch signal θ_d changes. A coupling between the PAH and the SAH systems is clearly present; it can introduce oscillation and also instability in the system. During a flocking simulation the aircraft spends most of its time following a target which flies at constant altitude; this effect can therefore not be neglected.

Designing the feedback gain of the speed control loop to have a deadbeat step response for the $\frac{\theta}{V_{Td}}$ transfer function, a safe design is conduced for the SAH module.

3.2.1 Actuators

Not only the aircraft is a physical system; also the actuator and the controller are realized using mechanic and electronic devices. Mathematic models for these systems can be then also introduced.

The controllers nowadays are realized with a digital computer, therefore they can be fairly modelled by a simple delay. If the dynamic of the system is significantly slower then the controller delay (as in the case of normal aircraft like the WOT4), the delay can be neglected without effecting the design.

The actuators used to deflect the aircraft control surface can be realized with different type of devices; mechanical devices, hydraulic devices or electrical devices (as it is for the WOT4). These devices mainly present a low-pass system behaviour; in this work they are approximated using simple lag networks with time constants shown in table 3.1.

With these constraints in mind the design can now be started.

Actuator	Time constant [s]
elevator	0.1
ailerons	0.05
rudder	0.05
engine	0.5

Table 3.1: Actuators time constant

3.3 Longitudinal autopilots

3.3.1 PAH

Figure 3.1 shows the schematic representation of the PAH control system.

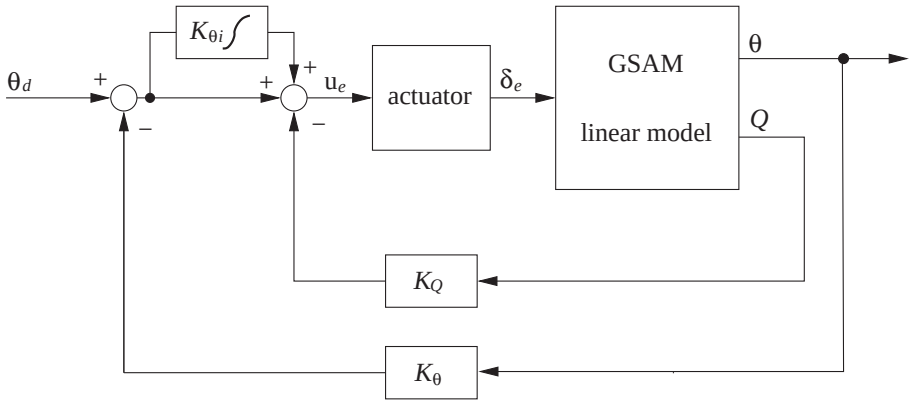

Figure 3.1: PAH control system

Two different control loops are used in the system, a θ feedback loop and a Q feedback loop. The main control loop is off course the θ control loop, it controls the pitch angle applying an elevator deflection if θ differs form θ_d. To have more control on the system dynamics, a derivative contribution is also accounted through the Q feedback inner loop.

The space-state representation of the longitudinal dynamic is obtained selecting the appropriate rows and columns from the $\mathcal{A}$ and $\mathcal{B}$ matrices of the system. The longitudinal state variables θ and Q are directly used as feedback; to have each one of them in the output vector $\mathbf{Y}$ the matrices $\mathcal{C}$ and $\mathcal{D}$ of the state-space formulation are chosen as shown below.

$$\mathcal{A} = \begin{array}{c} \begin{matrix} V_t & \alpha & \theta & Q \end{matrix} \\ \begin{bmatrix} -0.1538 & 3.8561 & -9.8100 & 0 \\ -0.0574 & -8.2042 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0.2386 & -87.184 & 0 & -14.535 \end{bmatrix} \end{array} \qquad \mathcal{B} = \begin{array}{c} \begin{matrix} \delta_e \end{matrix} \\ \begin{bmatrix} 0 \\ -0.7006 \\ 0 \\ -185.73 \end{bmatrix} \end{array}$$

$$\mathcal{C} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathcal{D} = [0]\,.$$

As can be noticed the dependence from h is not present anymore, in fact h does not give any contribution because a costant atmosphere model is assumed.

The δ_e to θ transfer function is

$$\frac{\theta}{\delta_e} = \frac{-185.73(s+7.8461)(s+0.1831)}{(s+0.0699 \pm j0.5708)(s+11.37 \pm j8.78)}. \tag{3.1}$$

The short period and the phughoid mode are clearly stable, but while the short period mode present a good damping ($\zeta_{sp} = 0.791$), the phughoid mode is very light damped ($\zeta_p = 0.121$). The actuator pole can modify the effect of the feedback, therefore must be accounted multiplying equation 3.1 by the transfer function

$$\frac{\delta_e}{u_e} = \frac{-10}{(s+10)}. \tag{3.2}$$

Note that in the numerator a negative sign is present; it is needed to have a positive pitch rate as consequence of a positive elevator deflection.

The effect of the θ feedback on the poles position can be inspected using the root locus sketch of the $\frac{\delta_e}{\theta}$ transfer function (fig. 3.2).

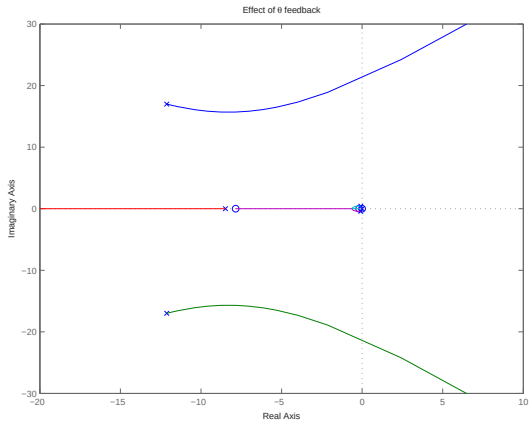

Figure 3.2: K_θ feedback effect

The θ feedback moves the short period pole towards the real axis increasing their damping; eventually they can terminate in the two remaining zeros. At the same time the feedback effect also reduces the damping one of the short period poles; a high feedback can eventually move them to the right half plane and cause instability. To reduce the steady state error, the dc gain must be as large as possible because

$$e_{\theta_{ss}}(\infty) = \frac{1 + G(0)(K_\theta - 1)}{1 + K_\theta G(0)}, \tag{3.3}$$

where $G(\omega)$ is the transfer function of the open loop system calculated in $s = 0$. The feedback gain must be then maximised.

A way to obtain the correct K_θ and K_Q gain is to fix K_θ and adjust K_Q for the best damping of the complex poles.

A compromise was obtained with $K_Q = 0.112$ and $K_\theta = 1.2$. With these feedback gain, the phughoid poles are in the real axis ($p_1 = 0.225$, $p_2 = 0.103$)

while the short period poles have a damping ratio $\zeta_{sp} = 0.479$. The short period damping is not excellent but was accepted because any further reduction of K_θ, lead to a system considered too slow. The low natural frequency of the short period complex pair ($\omega_{sp} = 1.79\ rad/s$) leads to a smooth oscillation considered acceptable.

Despite the attempt made to increase it, the dc gain of the system is really poor; and the resulting steady state error to a step input is

$$e_{\theta_{ss}}(\infty) = \frac{1 + G(0)(K_\theta - 1)}{1 + K_\theta G(0)} = 0.176. \tag{3.4}$$

An error of this magnitude can not be tolerated so an integrator (visible in fig. 3.1) is used in the forward path to control it.

The eventual steady state error $e_{ss} = \theta - \theta_d$ is integrated and used as additional positive feedback to the elevator. As effect of the integration the error contribute increase as more as the time pass, the position of the elevator is then adjusted consequently to force a null steady state error. This intuitive explanation is obviously in accordance with the control system theory. Adding the integrator the system is become a type-1 system, which presents as known a null steady state error to a step input.

To adjust the integration costant K_{θ_i} the step response was inspected. To obtain a fast response, a small overshoot was allowed and the value 1 was chosen for K_{θ_i}. The obtained step response is shown in figure 3.3.

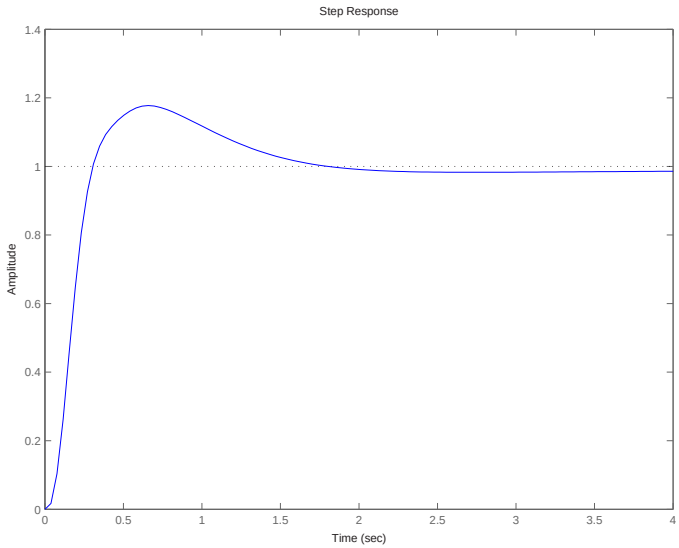

Figure 3.3: PAH module step response

The stability was also inspected using the Bode diagram (fig.3.3), the feedback gains chosen grant a satisfactory gain margin of 8.6dB @ 14.4 rad/s and a phase margin of 37.6 deg @ 7.3 rad/s.

3.3.2 SAH

The second control system developed for the longitudinal motion is the speed hold autopilot (SAH). As already introduced, when a target is present, a coupling

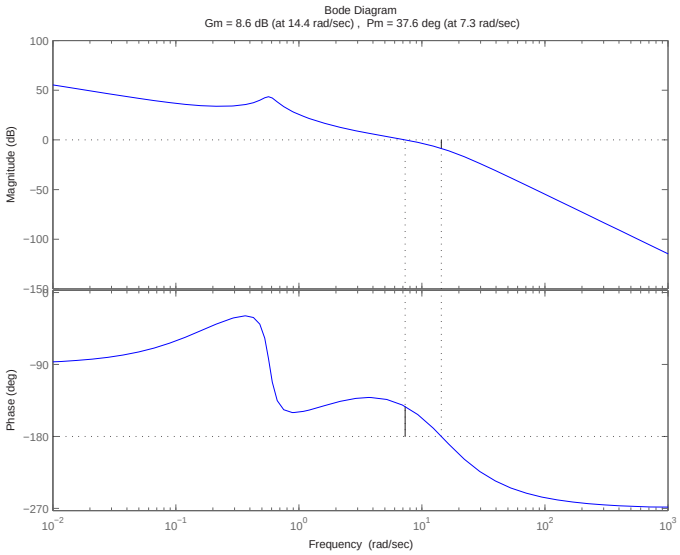

Figure 3.4: PAH Bode diagrams

between the speed and the pitch controls exists. To proceed with the design an expression which estimates this effect is formulated.

The extent of the coupling is dependent on the geometry of the situation. An example can help to obtain the desired equation.

Let us suppose that a target point T is flying at costant speed and costant altitude, followed by the aircraft A (fig.3.5) flying at the same speed. Let us also

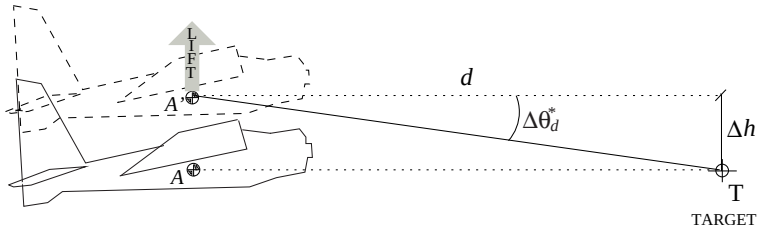

Figure 3.5: Target following, typical scenario

suppose that the angular displacement between the aircraft reference line and the target position is used as *correction* input θ_d^* for the PAH controller of the aircraft. The adjective correction wants to emphasise that the pitch input θ_d is obtained from

$$\theta_d = \theta_{ss} + \theta_d^*$$

where θ_{ss} is the level flight pitch.

In this situation, after an initial transient the aircraft will be flying at costant distance d from the target and θ_d^* will be null. A change of the aircraft thrust (to obtain a change of speed) leads to an altitude change Δh, therefore also to a change $(\Delta\theta_d^*)$ of θ_d^*. From geometrical consideration we have

$$\Delta\theta_d^* = \arctan\frac{\Delta h}{d}. \tag{3.5}$$

The coupling effect can be accounted adding the equation

$$\dot{\theta}_d^* = \frac{1}{K_{\theta_d d}} \frac{1}{1 + (\frac{\Delta h}{d})} \Delta h \tag{3.6}$$

obtained from a derivation of the equation 3.5 in the linear aircraft model. Note that this equation is added to the model only in this section with the only purpose of design properly the speed feedback loop. Extending the pitch controller scheme of figure 3.1, $\Delta\theta_d^*$ is integrated as an additional contribute to the feedback signal (fig. 3.6). Before going on with the design is worthwhile to stress that the

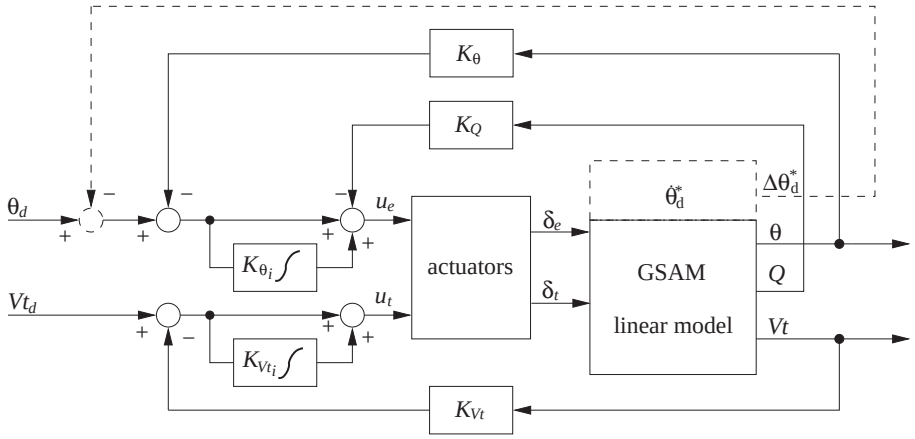

Figure 3.6: SAH, autopilot

situation examined is effectively the worst one. For a fixed Δh the magnitude of the coupling effect is the largest when the slope of the arctan function is the largest. Thus when $\Delta\theta_d$ is zero (the aircraft and its target are flying at the same altitude). If $h = 0$ the derivative $\dot{\theta}_d^*$ is also independent from the distance d.

During the design the attention is focused on the two transfer function

$$\frac{\theta}{V_{T_d}} = \frac{0.0127(s+12.49)(s+1.35)(s+0.833)(s+0.098)(s+1.014 \pm j3.59)(s+8.62 \pm j15.71)}{(s+8.62 \pm j15.70)(s+1.05 \pm j3.64)(s+12.49)(s+0.1)(s+0.95)(s+2)} \tag{3.7}$$

$$\frac{V_T}{V_{T_d}} = \frac{0.867(s+12.49)(s+0.96)(s+1.014 \pm j3.59)(s+8.62 \pm j15.71)}{(s+8.62 \pm j15.70)(s+1.05 \pm j3.64)(s+12.49)(s+0.1)(s+0.95)(s+2)} \tag{3.8}$$

obtained from the state-space representation of the system.

$$\mathcal{A} = \begin{array}{c} \begin{array}{rrrrr} V_t & \alpha & \theta & Q & \Delta\theta^* \end{array} \\ \begin{bmatrix} -0.1538 & 3.8561 & -9.8100 & 0 & 0 \\ -0.0574 & -8.2042 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0.2386 & -87.184 & 0 & -14.535 & 0 \\ 0 & -3.6791 & 3.6791 & 0 & 0 \end{bmatrix} \end{array}$$

$$\mathcal{B} = \begin{array}{c} \begin{array}{rr} \delta_e & \delta_t \end{array} \\ \begin{bmatrix} 0 & 0.433 \\ -7.006 & -0.0015 \\ 0 & 0 \\ -185.7 & 0.0063 \\ 0 & 0 \end{bmatrix} \end{array} \quad \mathcal{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathcal{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

The first is the proper function of the SAH autopilot,while the second is the transfer function of the coupling between V_T and θ. The open loop transfer function $\frac{V_T}{V_{T_d}}$ shows that all the complex poles are almost exactly cancelled. The feedback will then effect probably only a small number of poles.

The value of K_{V_T} is designed exploiting the root locus sketch (fig.3.7) of the transfer function 3.8. Most of the poles are not moved by the feedback while

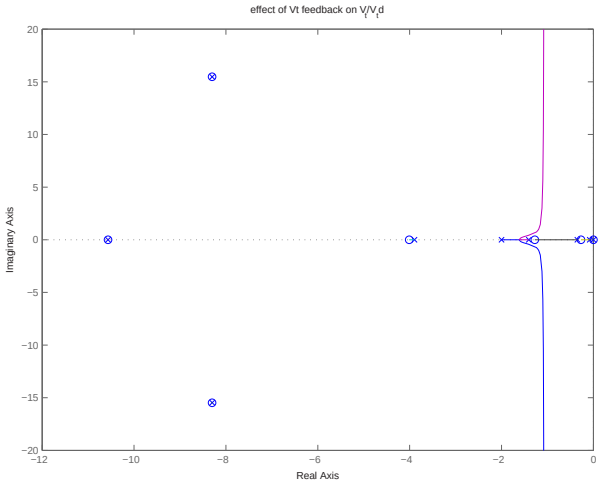

Figure 3.7: SAH, V_T feedback effect

$p_1 = -2$ and $p_2 = -1.41$ are moved one towards the other. If the feedback gain is higher then 0.374 they depart from the real axis forming a new complex pair. The poles in -0.348 and -0.073 moves on the real axis in the left direction. If the gain is high enough they can cancel out respectively with the zeros in -0.77 and in -1.28. To guarantee a smooth response a damp value of 0.9 for the new complex pair is obtained imposing K_{V_T} equal to 1.36.

As was for the PAH controller also in this case the dc gain of the system is poor, and the resulting steady state error is unacceptable ($e_{V_{Tss}} = 0.16$). As done in the PAH autopilot an integrator is introduced in the system. A large value for $K_{V_{T_i}}$ allows a fast removal of the steady state error; but the position of the zero of the integral proportional compensator can also change the effect of the feedback. To avoid this, the zero must be placed near the pole in -0.07 to cancel out it. Therefore it was placed in $s = -0.1$ ($K_{V_{T_i}} = 0.1$). The V_T to V_{T_d} step response obtained is shown in figure 3.8.

The poles of the $\frac{\theta}{V_{T_d}}$ transfer function are exactly the same already examined therefore also the feedback effect on them is the same. The feedback realized will then also avoid oscillations in the V_{T_d} to θ step response. As expected figure 3.9 shows that for a steady state level flight with a higher speed,a lower angle of attack is needed (therefore a low pitch angle) is needed.

3.4 Lateral autopilots

The main function of this autopilot is to provide a yaw auto hold system (YAH).

The autopilot has to apply the correct banking needed to reach an then mantain the desired ψ_d angle. The system developed is shown in figure 3.10.

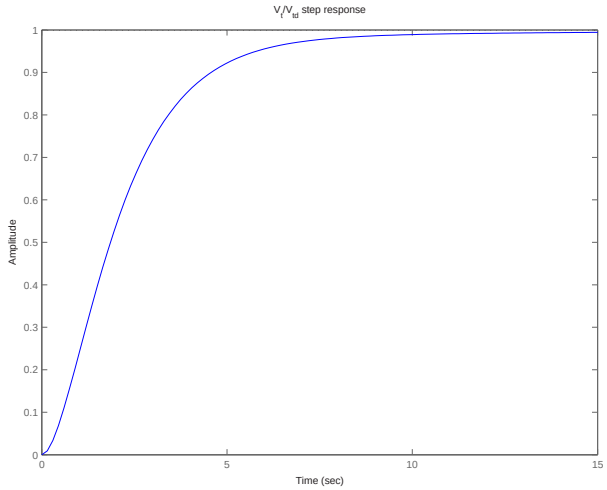

Figure 3.8: SAH, $\frac{V_T}{V_{T_d}}$ step response

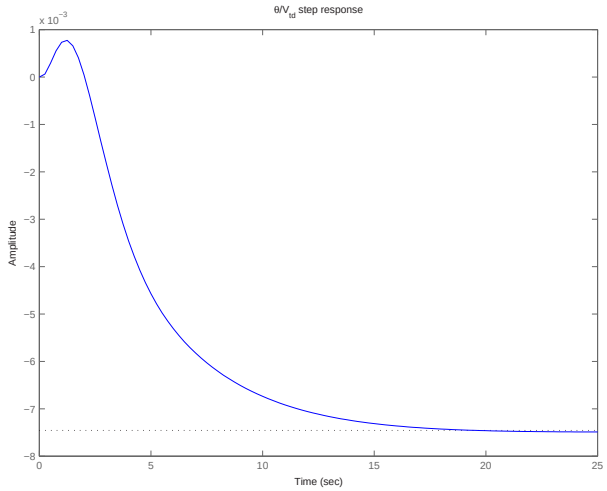

Figure 3.9: SAH, $\frac{\theta}{V_{T_d}}$ step response

To control properly the banking angle a roll hold autopilot (RHA) is designed and exploited as inner loop. In the RHA the roll rate is used as derivative feedback in addition to the proportional feedback, with the aim of improving the control capabilities. To limit the sideslip and to control the Dutch Roll mode, a roll damper is also then realized using the yaw rate as feedback for the rudder. The rudder in this way generates a yawing moment to counteract the yawing motion coming from the Dutch Roll mode. A simple feedback from the yaw rate is not the best choice. For example during a steady state turn, the yaw rate has a constant value which differs from zero, therefore a constant rudder deflection would be applied even if is no longer necessary; the Dutch Roll in fact most likely already died. As suggested in [17], using a first order high pass filter a derivative effect for the yaw rate can be obtained , so the rudder control effect will be produced only during a variation of the yaw rate as needed. In the aeronautic field such type of filter is commonly known as washout filter.

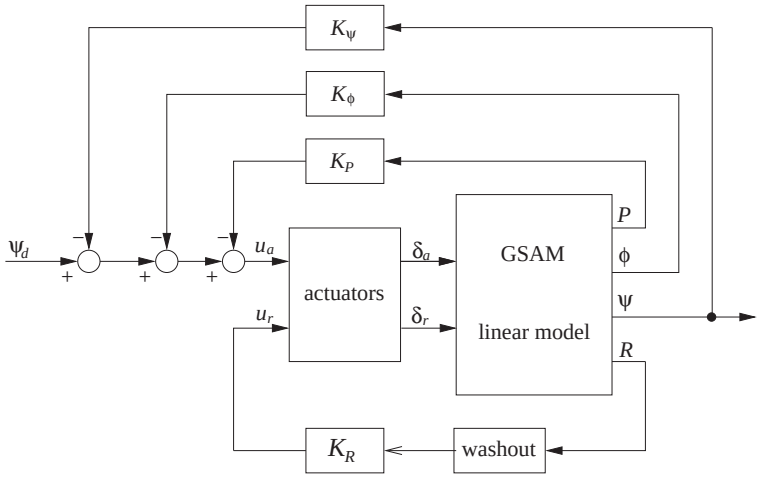

Figure 3.10: YAH control system

The system state-space representation is given by the lateral subset of the Jacobian matrices.

$$\mathcal{A} = \begin{array}{c} \begin{matrix} \beta & \phi & \psi & P & R \end{matrix} \\ \begin{bmatrix} -0.9108 & 0.5322 & 0 & 0.0650 & -0.9979 \\ 0 & 0 & 0 & 1 & 0.0651 \\ 0 & 0 & 0 & 0 & 1.0021 \\ -6.8983 & 0 & 0 & -3.2340 & 0.1188 \\ 17.173 & 0 & 0 & -0.1683 & -0.9971 \end{bmatrix} \end{array}$$

$$\mathcal{B} = \begin{array}{c} \begin{matrix} \delta_a & \delta_r \end{matrix} \\ \begin{bmatrix} 0 & 0.3258 \\ 0 & 0 \\ 0 & 0 \\ -62.982 & 2.2680 \\ -8.1992 & -14.1617 \end{bmatrix} \end{array} \quad \mathcal{C} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathcal{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

3.4.1 RAH

Since the roll and yaw motion are not independent the MIMO design of the RAH and the yaw damper are carried out together first. The three transfer function below are studied:

$$\frac{P}{u_a} = \frac{125.9s(s+20)(s-0.329)(s+0.97 \pm j4.25)}{s(s+0.887 \pm j4.22)(s+0.0097)(s+3.45)(s+20)^2} \tag{3.9}$$

$$\frac{\phi}{u_a} = \frac{127.1s(s+20)(s+0.96 \pm j4.237)}{s(s+0.887 \pm j4.22)(s+0.0097)(s+3.45)(s+20)^2} \tag{3.10}$$

$$\frac{R}{u_r} = \frac{283.2(s+20)(s+0.189 \pm j0.7834)}{s(s+0.887 \pm j4.22)(s+0.0097)(s+3.45)(s+20)^2} \tag{3.11}$$

According to [16], the roll damping loop is closed first because less critical. The figure 3.11 shows the effect of the P feedback. The first effect (visible in the detail) is to move the Dutch Roll poles towards the respective zeros, while the second is

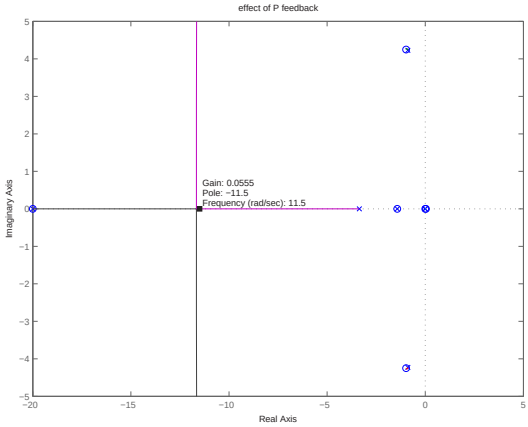

Figure 3.11: RAH, P feedback effect

to move the subsidence pole towards the spiral pole and vice versa. Eventually if the feedback gain is larger then 0.0555 they will depart from the real axis and form a new complex pair. To have a fast deadbeat response a feedback gain of 0.05 was chosen.

The second step is now to design the yaw rate feedback. The main effect of the R feedback (fig.3.12) is to move the Dutch Roll poles towards the open

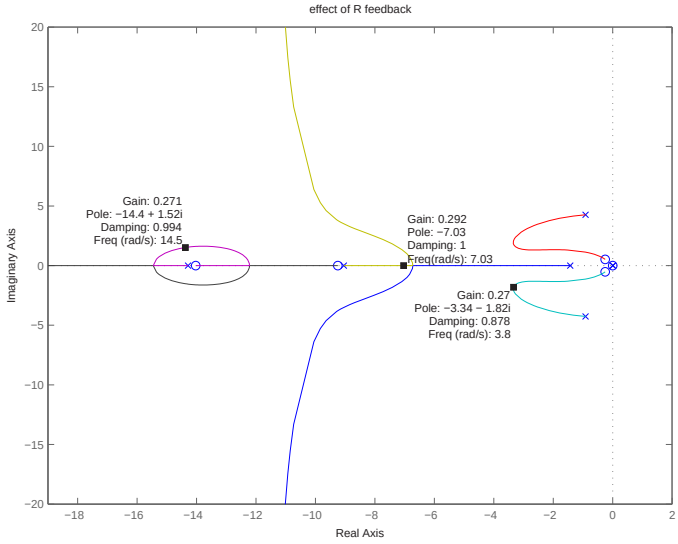

Figure 3.12: RAH, R feedback effect

loop zeros, with the desired effect of improving the damping coefficient of this complex pair. The feedback can also lead to the formation of a complex pair from the combination of the washout and the spiral pole if the gain is higher then 0.292. It is obviously preferable to avoid the introduction of further complex pair, therefore a value of 0.297 was chosen for K_R. With this gain the Dutch Roll poles are not completely cancelled by the open loop poles, but the position of the washout pole and its attractive action can be exploited to obtain a good damping. With a feedback gain of 0.277 and a washout time constant of 0.7 s a satisfactory damping of 0.88 can be obtained.

In the left side of the sketch, can be noticed also a new complex pair formed by the spiral pole and the aileron actuator pole. Its distance from the imaginary axis and its large damping (0.99) make its action on the system negligible.

The figure 3.13 shows the root locus plot of the ϕ feedback loop. The Dutch

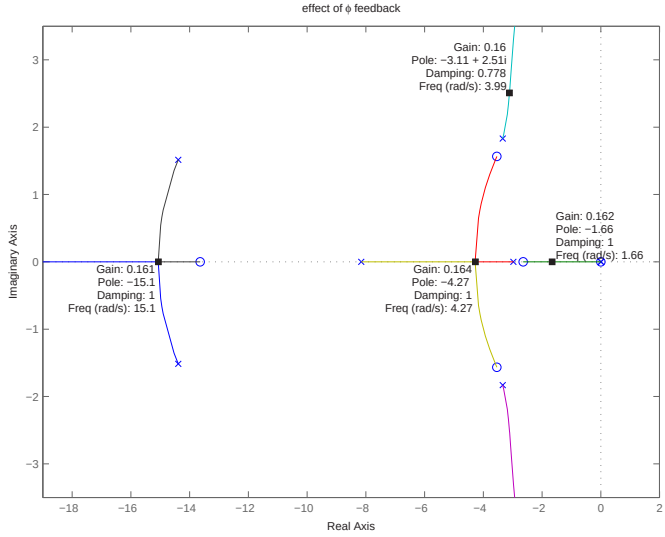

Figure 3.13: RAH, ϕ feedback effect

Roll poles are moved away from the respective zeros, and a K_ϕ value higher than 0.161 forms a complex pair from the washout and the aileron actuator poles. The ϕ feedback also moves the subsidence pole, in the left direction. A feedback gain higher than 0.160 will also overdamp the complex pair previously rose from the spiral pole and the actuator pole because of the roll rate feedback. A value of $K_\phi = 0.16$ is thus the preferable choice.

The response of the obtained system to a ϕ step input is reported in figure 3.14. As confirmed by the figure the integrative effect of the subsidence pole near

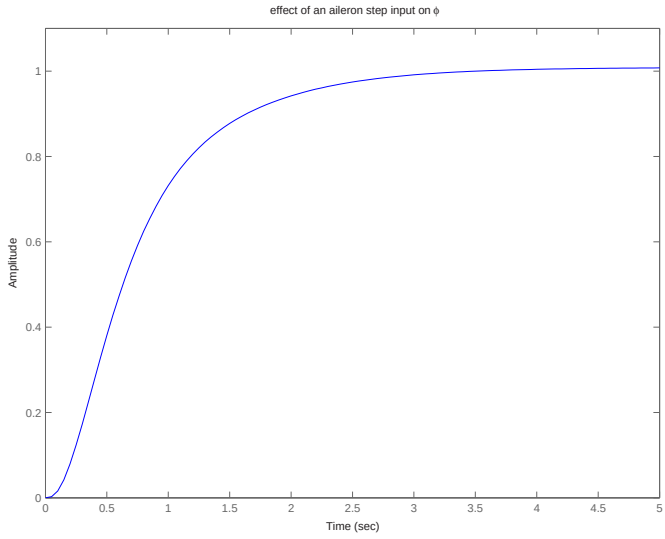

Figure 3.14: RAH, ϕ step response

to the origin, rise the dc gain, the steady state error for a step input is therefore low,

$$e_{\phi_{ss}} = \frac{1 + G(0)(K_\phi - 1)}{1 + G(0)K_\phi} = 0.0067.$$

3.4.2 YAH

Using the RHA just designed, the YAH autopilot can be obtained simply adding a ϕ feedback loop to the roll input. With this configuration, to perform a turn, the aircraft is initially banked to an angle proportional to the needed change of heading, then, when the requested heading is approached, the banking is reduced and the level flight is restored. This is the fastest way to perform the requested manouvre but, is not always the correct way.

Since the peak value of the banking angle is proportional to the commanded heading variation, this control scheme can lead to undesirable large bank angle. The bank angle must thus be limited. As explained in [18] for a steady level coordinate turn, the turn radius can be calculated by

$$R = \frac{V^2}{g \tan \phi}. \tag{3.12}$$

Later in section 3.4.3, is shown that the loss of altitude during a turn is also related to the magnitude of the bank angle. The presence of the tan function at the denominator of the equation 3.12 suggests that good turn performance can be obtained with fair small banking angle. A set of non linear simulations showed a maximum angle of 30° as a good trade off between the presented effects.

A further validation of the necessity of clamping the bank angle comes directly from the flight manouvre of a human pilot. A human pilot usually performs small turn and large turn in two different manners. When performing a small heading change, he simply banks the aircraft (to an extent suggested by experience and visual clues), and then rapidly recovers the steady state level flight. Conversely for a large turn, he banks the aircraft to a certain angle (suggested also by his experience), performs almost all the turn more or less at constant turn speed, and finally recovers the level flight. With the introduction of a maximum bank angle, the YAH controller will imitate this behaviour.

The open loop transfer function of the system is

$$\frac{\psi}{\psi_d} = \frac{164.33(s + 1.42)(s + 20)(s + 4.32)(s + 0.73 \pm j4.07)}{s(s + 14.74)(s + 1.72)(s + 3.09 \pm j2.59)(s + 4.23 \pm j0.37)}. \tag{3.13}$$

To design K_ψ once again the root locus sketch of the ψ feedback is used. A closed look to the root locus near the origin (fig.3.15), reveals that a feedback gain larger then 0.011 moves the spiral pole and the pole in -1.64 away from the real axis to form a complex pair. High values of K_ϕ can also move this complex pair to the right hand plane and lead to an instability. Due to is short distance from the imaginary axis, this complex pair is clearly the major contribute to the system dynamics. To reach a good compromise between the time response and the oscillation characteristic, a gain of 0.12 was chosen. The resulting damping of this complex pair is 0.81. With this K_ϕ gain also the Dutch roll pole are moved away but their damping (0.76) is still satisfactory. Not visualised in the figure,the

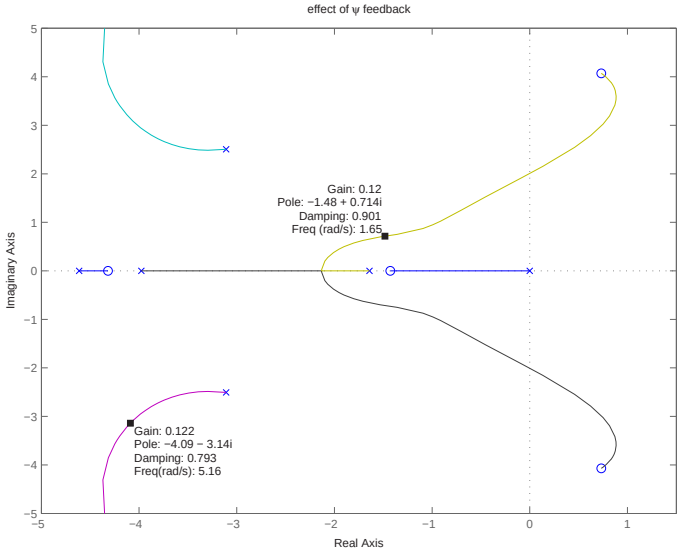

Figure 3.15: YAH, effect of ψ feedback

complex pair previously raised because of the K_R feedback has a damping of 0.99 almost independent from K_ϕ.

As conclusion of the design, the step response of β and ψ to a heading step input are shown in figures 3.16 and 3.17.

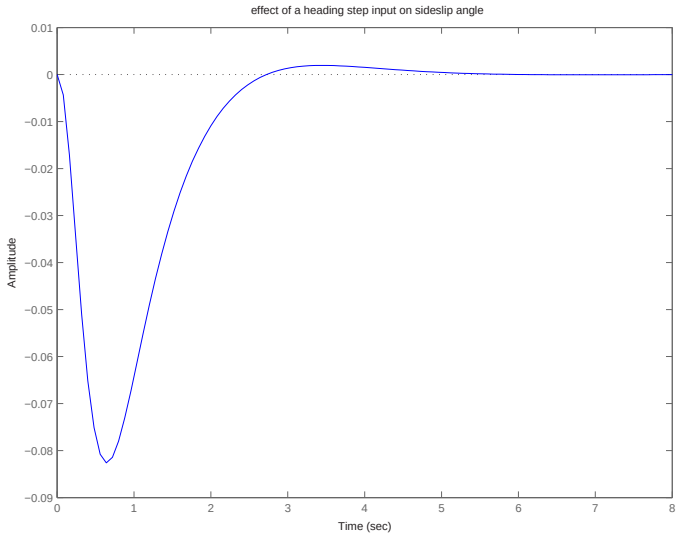

Figure 3.16: YAH, β response to a ψ_d step input

As expected the ψ step response does not show overshoot, the steady state error is zero because a pole in the origin is present in the transfer function 3.13.

Due to the fact that the Dutch Roll mode is not completely damped, the ψ_d input step still produces an initial perturbation of the sideslip angle. The small amplitude of the perturbation and its good damping allow to consider the result satisfactory. Additional validation will be given with the non linear simulation.

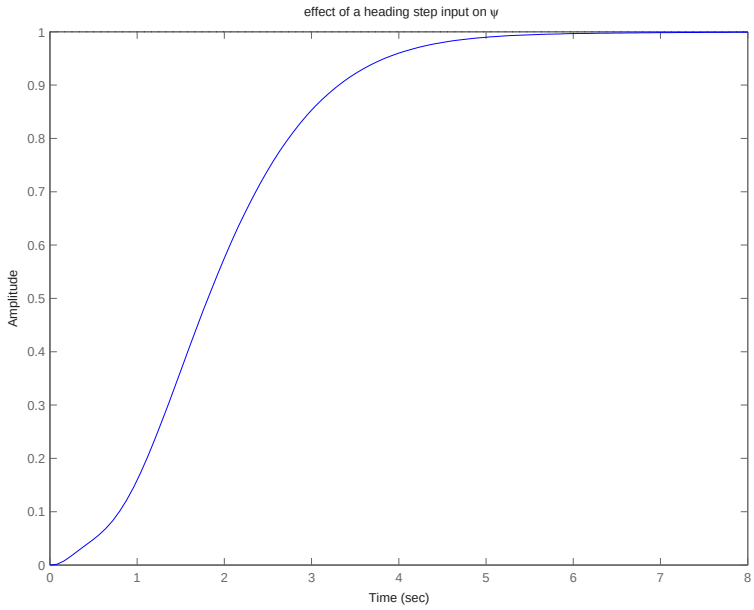

Figure 3.17: YAH, ψ step response

3.4.3 Level turns

During a turn a bank angle different from zero, changes the vector sum of lift and gravity force. As figure 3.18 shows, since the lift force is always orthogonal

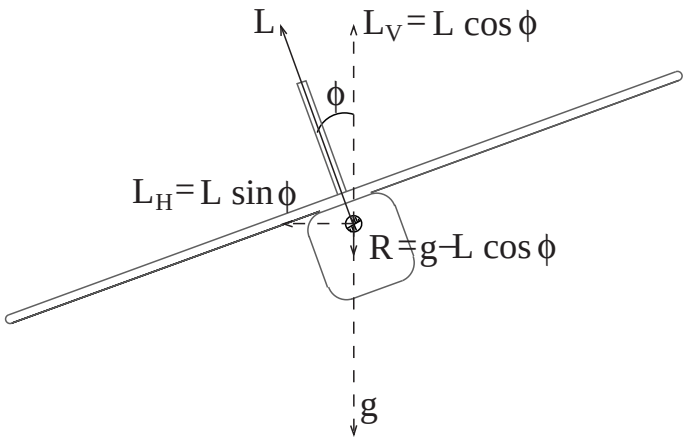

Figure 3.18: Lift loss during a turn

to the wings, during a turn only a reduced part of it

$$L_V = L\cos\phi$$

counteracts the weight force. While the horizontal component of the lift force

$$L_H = L\sin\phi$$

is responsible of the turn motion; the vertical resultant

$$R = g - L\cos\phi$$

causes a loss of altitude. Several schemes to compensate this effect can be found in literature, the simplest are based on an additional contribute to the elevator proportional to the lift loss

$$\Delta L = L(1 - \cos\phi).$$

The total altitude loss is proportional to the bank angle and to the time duration of the turning. The duration of the turn is determined by the magnitude of the performed heading change. In the present situation the bank angle is limited to 30° by the PAH. This limitation strongly reduces the altitude loss. Moreover during a flocking flight, the PAH is working, it will counteract the altitude loss with an elevator input as already noticed in the section 3.3. The altitude loss is then definitely negligible.

As example in figure 3.19 is reported the time evolution of the altitude during a turn of 60° (started at $t = 1s$), while the aircraft is following a target from a distance of 10 meters.

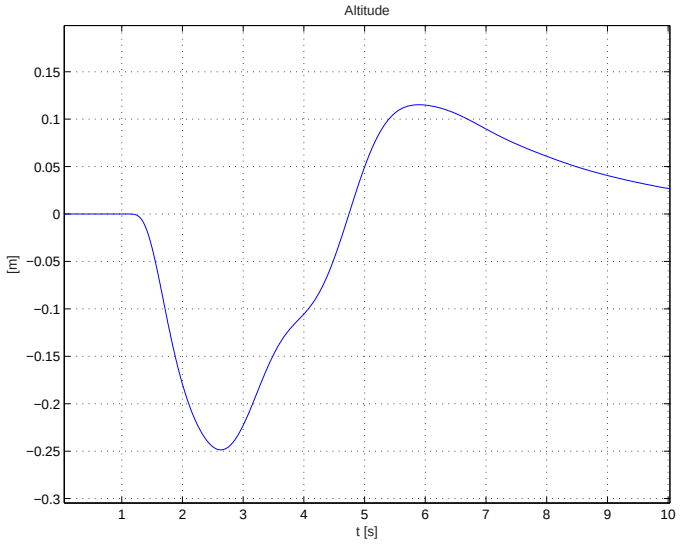

Figure 3.19: Altitude loss during a 60° turn

The MATLAB code of the resulting lateral and longitudinal controller is as usual included in appendix B.5.

3.5 Airspeed dependence

In the previous sections all the feedback parameters were determined for the linear model obtained in the equilibrium point characterized by a steady state flight at 18.39 m/s.

As already explained in section 2.8 the space state model is directly dependent from the flight condition. The dependence of the dimensionless aerodynamic coefficients from the airspeed is neglected but all the dimensional aerodynamic forces and moments are hardly dependent from the airspeed. The positions of the poles are conditioned by the airspeed, therefore also the control system parameters must vary with the airspeed.

Following the approach proposed in [17], using consideration similar to those presented in section 3.3 and 3.4 all the control system was redesigned for several airspeeds. To have a good evolution of the airspeed dependences in the flight envelope all the equilibrium points of table 2.6 here reported for completeness were inspected.

$\mathbf{V_{t_e}}$ $[m/s]$	$\theta = \alpha$ $[rad]$	$\delta_\mathbf{e}$ $[rad]$	$\delta_\mathbf{a}$ $[rad]$	$\delta_\mathbf{r}$ $[rad]$	$\delta_\mathbf{t}$ $[N]$
11	0.1871	0.078	0	0	2.34
12	0.157	0.059	0	0	2.31
14	0.115	0.032	0	0	2.44
15	0.099	0.022	0	0	2.57
18.39	0.065	0	0	0	3.26
22	0.044	-0.013	0	0	4.32
25	0.033	-0.020	0	0	5.42
28	0.026	-0.025	0	0	6.69
30	0.022	-0.027	0	0	7.63
32	0.019	-0.029	0	0	8.64
33	0.018	-0.030	0	0	9.18

Table 3.2: Equilibrium points

The feedback gains K_θ, K_Q and K_{θ_i} present in the PAH autopilot, show to be fairly independent from the airspeed. Similar consideration can be carried out on K_{V_T}. Conversely $K_{V_{T_i}}$ (the speed error integrator gain) is clearly airspeed dependent (fig.3.20). By means of a minimum square fitting, a third order equation of

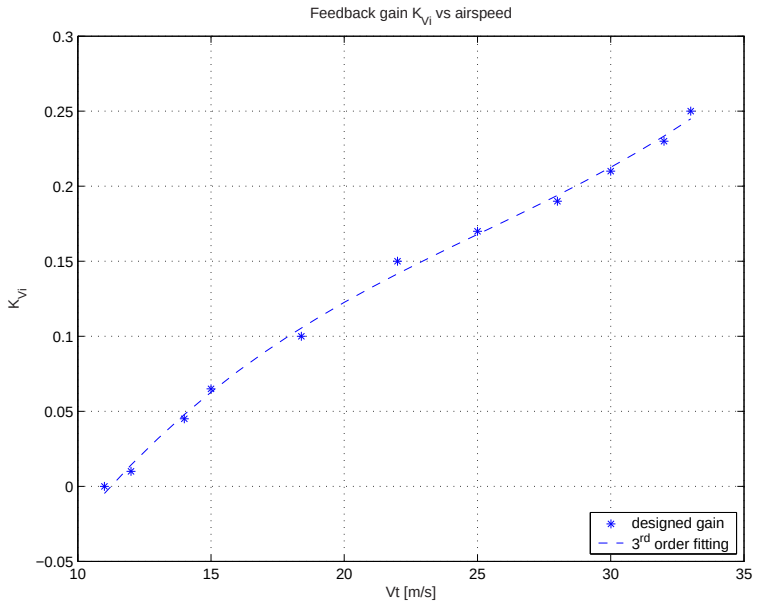

Figure 3.20: $K_{V_{T_i}}$ vs airspeed

the $K_{V_{T_i}}$ dependence from airspeed was obtained.

$$K_{V_{T_i}} = 2.89\,10^{-5}\,V^3 - 2.145\,10^{-3}\,V^2 + 6.096\,10^{-2}\,V - 0.467. \tag{3.14}$$

In the same manner 4^{th} order relation were obtained for the feedback gains K_psi,K_P and K_R.

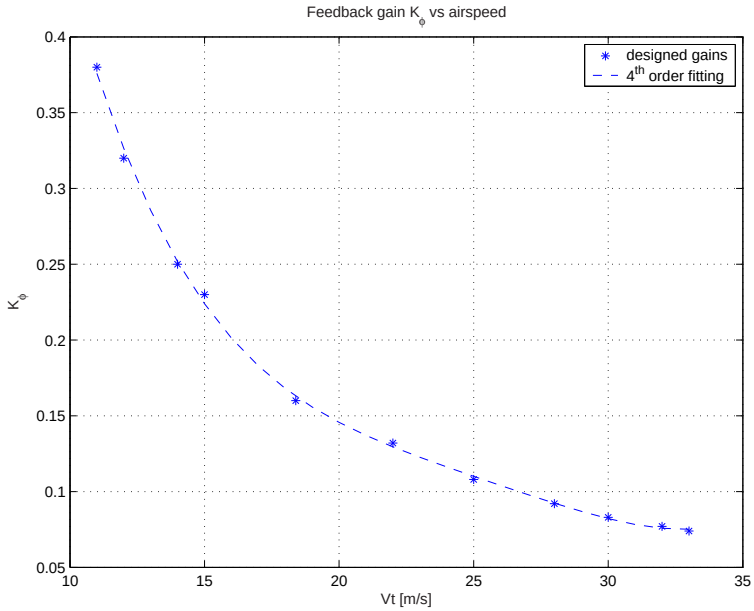

Figure 3.21: K_ϕ vs airspeed

$$K_\phi = 3.67\,10^{-6}\,V^4 - 3.77\,10^{-4}\,V^3 + 1.46\,10^{-2}\,V^2 - 0.257\,V + 1.89. \quad (3.15)$$

$$K_P = 2.04\,10^{-6}\,V^4 - 2.16\,10^{-4}\,V^3 + 8.60\,10^{-3}\,V^2 - 0.155\,V + 1.10. \quad (3.16)$$

$$K_R = 8.18\,10^{-6}\,V^4 - 8.03\,10^{-4}\,V^3 + 2.93\,10^{-2}\,V^2 - 0.477\,V + 3.21. \quad (3.17)$$

The remaining gain K_ψ, is also almost speed independent.

The equations 3.14÷3.17 were directly integrated in the control algorithm; every control step the MATLAB function

```
[Kvti,Kpsi,Kp,Kr]=KKKK(Vt)
```

computes the feedback gains suitable for the current flight condition.

3.6 Controller with limiters

In a real scenario not only the aircraft exhibit a non linear behaviour, but also the control system is not linear. The non linearity of the controller are results of the limitation of mechanic an electronic devices that realize the controller.

For the aim of this work, the non linearity are accounted using two type of limitation:

- control surface deflection limitations;
- control variable limitations.

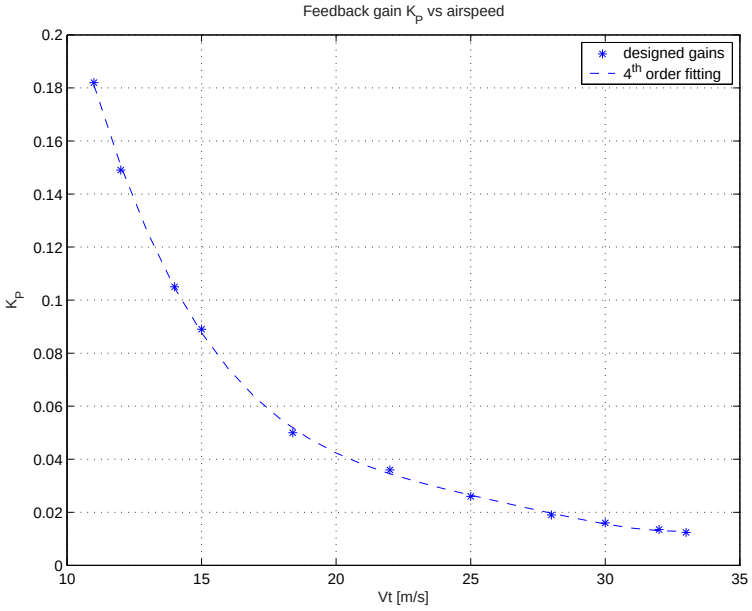

Figure 3.22: K_P vs airspeed

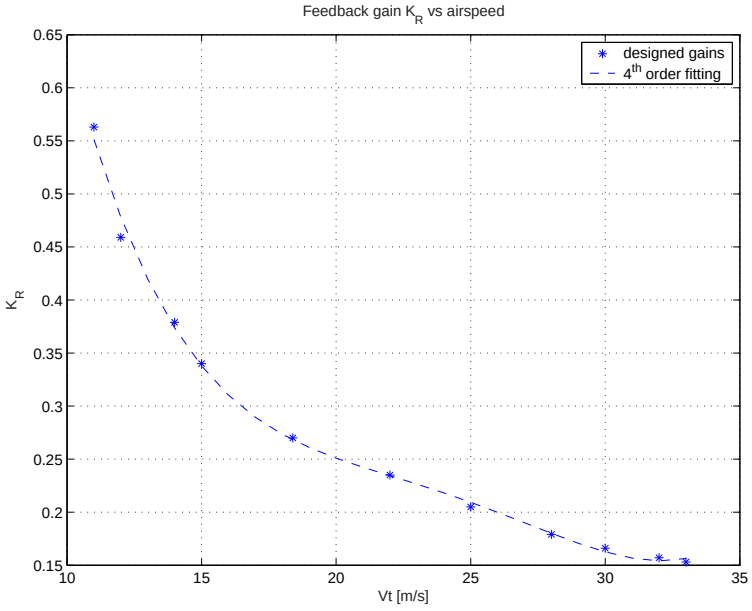

Figure 3.23: K_R vs airspeed

	δ_e	δ_a	δ_r	δ_t
range value	$\pm 25°$	$\pm 20°$	$\pm 30°$	$0 \div 9.8N$
derivative range value	$\pm 60°/s$	$\pm 80°/s$	$\pm 120°/s$	$\pm 50N/s$

Table 3.3: Controller limits

In table 3.6 are shown the limits used.

The limitation of the control derivative is realized simply clamping every derivative when a value out of the limit occurs. With the control variables a different strategy is adopted, when the variables reach the limit its derivative is accounted only if it is on the direction that takes the variable off the limit, otherwise the derivative is considered zero.

In the case of controller with integrator in the forward path, the clamping of the integrator input must also be realized simultaneously to the variable clamping. Otherwise the difference between the commanded and the obtained control value is continuosly integrated, and the saturation of the controller is reached.

The controller limiters are directly implemented in the routine responsiable of the actuator dynamic B.5.

3.7 Non linear simulation

To validate the controller designed exploiting the linear model, two simulations conduced with the non linear model and non linear controller are now presented.

In the first simulation a turn and a change of speed are performed. To inspect the "altitude" control effect performed by the PAH, the aircraft is supposed to be following a target flying in its same direction just 10 m afterwards. The pitch control is then generated with the modality explained in section 3.3.

The manoeuvres performed during the 20 s simulation are the sequent:

- $t = 0$ steady state flight $V_T = 18.39\text{m/s}$;
- $t = 2s$ turn of 90°;
- $t = 10s$ speed reduced to 14 m/s.

The ground trajectory (fig. 3.25) shows of course only the 90° turn, which exhibit a smooth change of heading and no overshoot as desired. Due to the large turn commanded, the bank angle is larger than the maximum, therefore ϕ_d is limited by the control system (fig. 3.27). The turn is not perfectly coordinated, a sideslip motion is still present (fig. 3.26) but the control action of the rudder (fig. 3.24) damps properly the Dutch Roll mode. When the turn is started the pitch angle exhibit a small decrease due to the pitching down effect given by the banking. The PAH module rapidly counteracts this effect increasing the aileron deflection, and also compensates the altitude loss with an additional elevator deflection.

At $t = 10s$, to reduce the cruise speed, the SAH module initially reduces to zero the thrust, and then gradually raise it to the cruise value of 2.37 N. When the airspeed is decreased (fig. 3.28), the PAH module react to the altitude loss increasing the pitch angle virtually in every instant a steady state flight is realized . The final value of 6.34° lead to a steady state flight at the commanded speed of 14 m/s.

The hypothesis of decoupling are verified so as expected, the second manouvre of speed change does not have any influence on the lateral variables ϕ,ψ,β.

In the second simulation a direct pitch manouvre is tested, therefore no target are present; a direct control of θ_d is considered:

- $t = 0$ steady state flight $V_T = 18.39\text{m/s}$;

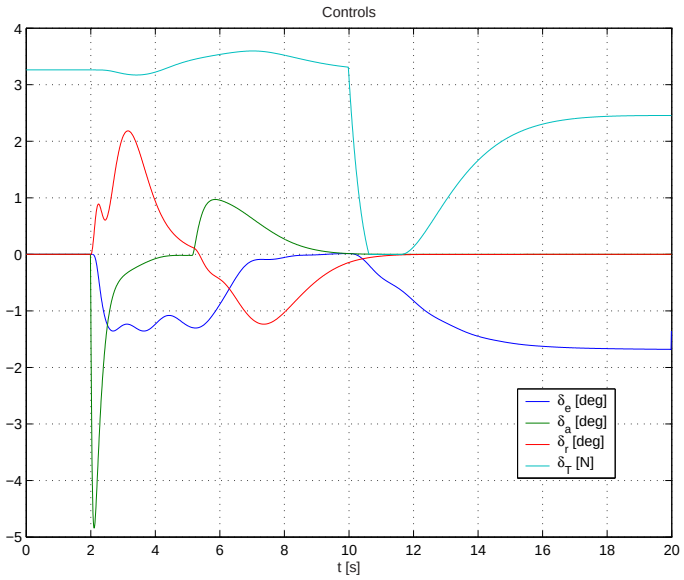

Figure 3.24: Turn and airspeed change manoeuvres, control inputs variation

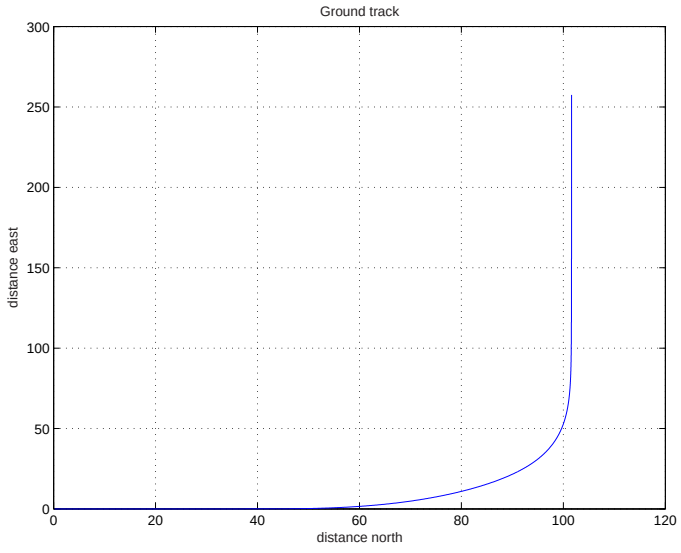

Figure 3.25: Turn and airspeed change manoeuvres, ground track

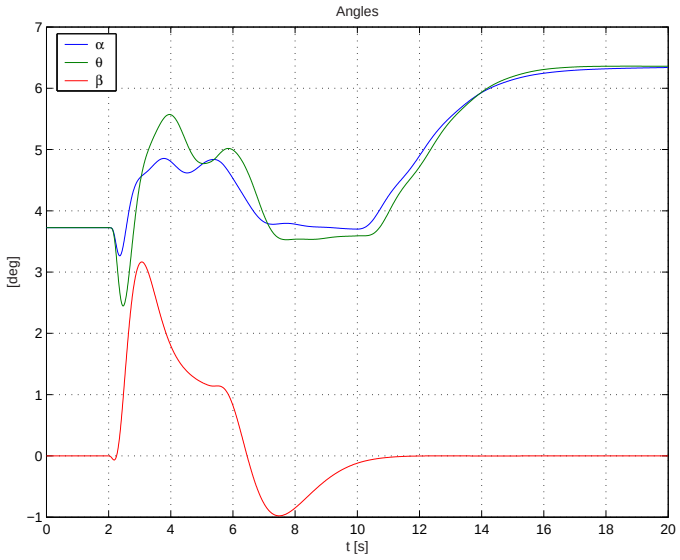

Figure 3.26: Turn and airspeed change manoeuvres, α,θ,β variation

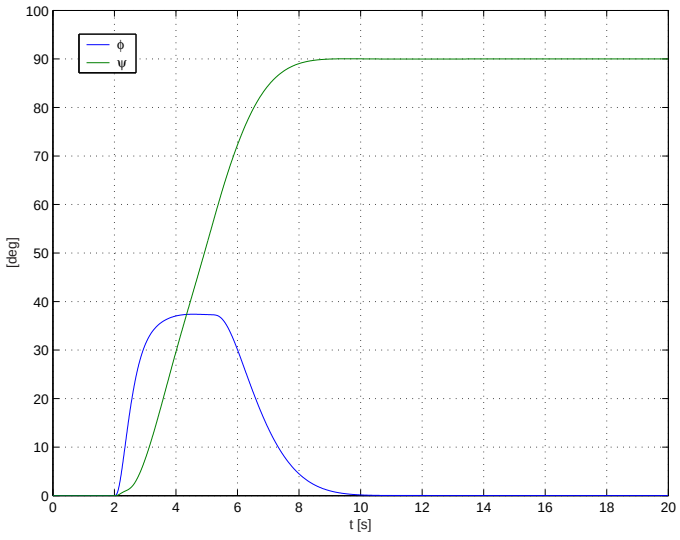

Figure 3.27: Turn and airspeed change manoeuvres, ϕ and ψ variation

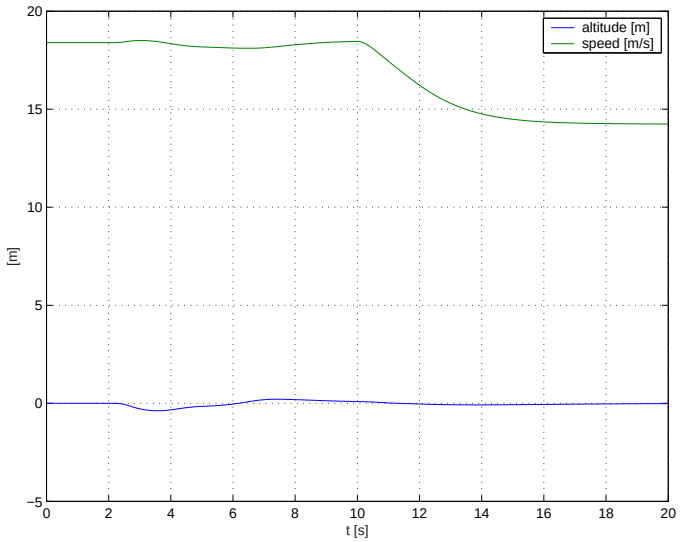

Figure 3.28: Turn and airspeed change manoeuvres, altitude and airspeed variation

- $t = 2s$ pitch rais to 9.45°.

A very fast response is exhibited (fig. 3.31) with a small overshoot. The angle of attack initially rise as effect of the strong elevator input, but then it stabilizes to give a constant speed flight with the commanded pitch angle. The change of α changes the drag and therefore the airspeed but the SAH increase the trust (fig. 3.29) to maintain the cruise speed of 18.39 m/s. The difference between the pitch

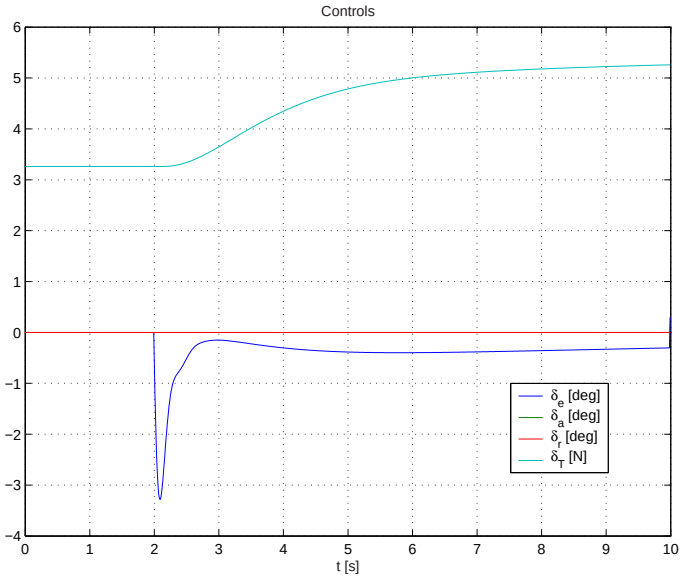

Figure 3.29: Pitch change, control inputs variation

angle and the angle of attack is called path angle. The aircraft is proceeding in a steady state flight with constant ascending speed (fig. 3.30), and the path angle is actually the angular displacement between the ground and its forward velocity.

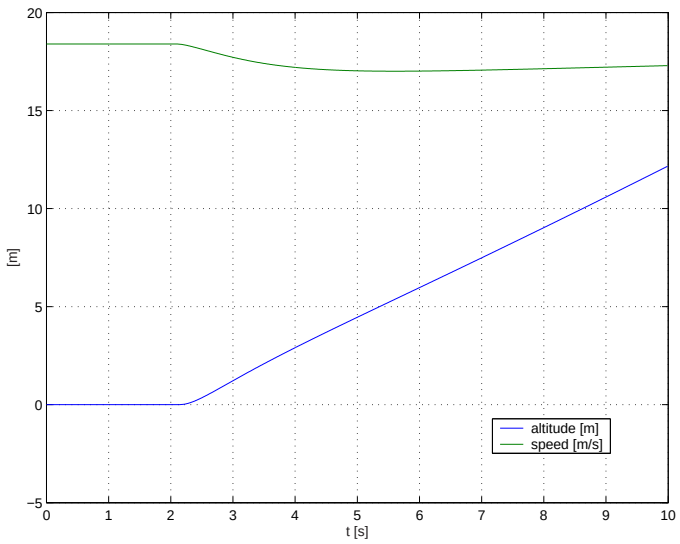

Figure 3.30: Pitch change, altitude and airspeed variation

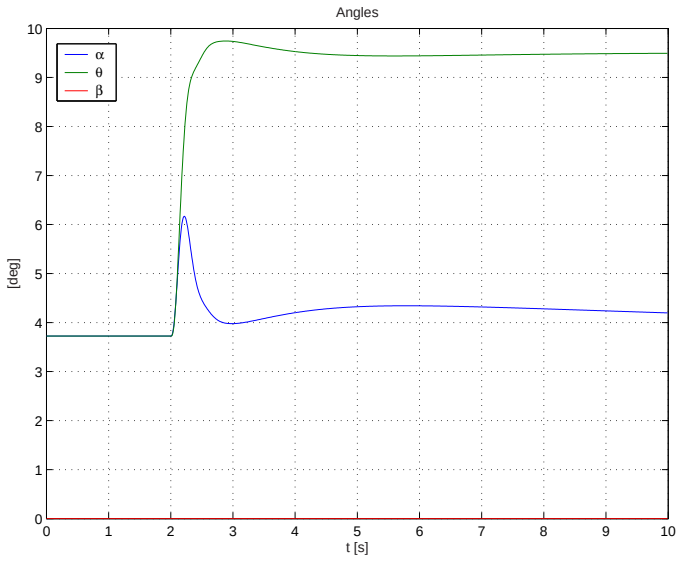

Figure 3.31: Pitch change, α,θ,β variation

3.8 Software library

The Matlab interpreted code used through these two chapters, particularly suitable during the design of the algorithms, is definitely slower than a compiled code.

In the next chapter and in other occasion during the development of the gridswarm project, the aircraft model and the control system created here, will be intensively used. It is thus really worthwhile to rewrite all the algorithms in C code.

To improve the usability of the code all the function are enclosed in a single library (*gsaml.a*); a strong effort was made to simplify the use as much as possible. For obvious motivation of space all the code is not reported in appendix, it will be downloadable from the gridswarm website.

The functions present in the library are now briefly presented, a more detailed description is present on the source code.

Library header file(`gsaml.h`)

Contains the declarations of all the aircraft and control system parameters. Contains also the definition of the structure `gsaml_boid_ds` which represent a container for all the state variable of a single aircraft.

```
struct gsaml_boid_ds {
  double *x;     // state vector Vt a b theta phi psi Q P R Pn Pe h
  double *n;     // actuator state vector
  double *w;     // controller state vector
  double *c;     // controller clamping state vector
  double sft;    // steady state thrust
  double MAXPHI; // max bank angle
};
```

Functions

Using a Java-like definition the following two functions should be declared as *public*; these are in fact the only two functions generally needed to use the library.

`gsaml_boid_ds* gsaml_new_boid (double pn,double pe,double h,double psi,int Vt,float MPHI)`
Initialises a new aircraft and returns its pointer.

The aircraft the control system state variables are initialised to the values associated with a steady state level flight at the cruise speed `Vt`. The value of `Vt` must be chosen in within the set `{11,12,14,15,18,22,25,28,30,32}`. The position and orientation of the aircraft are determined by `pn,pe,h,psi`, while `MPHI` determines its maximum bank angle.

Designed to be directly invoked by the user program.

`void gsaml_boid(gsaml_boid_ds *theboid,double *d,double dt)`
Main function.

Computes the new values of all the control system, actuator and aircraft state variables at the time $t + \Delta t$. All the subsystem autopilots, actuators,

aircraft model and washout filters are accounted. The present values of the state variables are passed to the function using the proper structure `theboid`. The vector `d` defines the control surface deflections and thrust to be used as input for the control system. The variable `dt` defines the integration time step. Designed to be directly invoked by the user program.

The remaining functions with a java-like definition should be declared as *private*; because directly invoked by the two previous ones. Generally these do not need to be directly used.

`void gsaml_gsam(double *x,double *u,double *xdot)`
Aircraft model routine.

Computes the new value of the aircraft state vector derivative `xdot` using the aircraft state vector `x` and the input vector `u`.

It is not designed to be directly invoked by the user program, in fact it is usually invoked by `gsaml_igsam`

`void gsaml_igsam(double *x,double *in,double dt)`
Calls `gsaml_rk` to integrate the values returned by `gsaml_gsam`over the time `dt`.

It is not designed to be directly invoked by the user program, in fact it is usually invoked by `gsaml_boid`.

`void gsaml_ad(double *n,double *uc,double *ndot)`
Actuator dynamic routine.

Computes the new value of the actuator state vector derivative `ndot` using the actuator state vector `n` and the control vector `uc`.

It is not designed to be directly invoked by the user program, in fact it is usually invoked by `gsaml_iad`

`void gsaml_iad(double *in,double *c,double *n,double *u,double dt)`
Actuator integration routine. Integrates the actuator derivative over the time `dt` using the **gsaml_rk** routine.

It is not designed to be directly invoked by the user program, in fact it is usually invoked by `gsaml_boid`.

`void gsaml_wi(double *w,double *z,double *wdot)`
Washout filter routine.

Approximates the derivative of the roll angle using a low pass function.

It is not designed to be directly invoked by the user program, in fact it is usually invoked by `gsaml_ics`

`void gsaml_ics(double *x,double *c,double *d,double s,double M,`
`double *w,double *u,double dt)`
Control system routine.

Compute from the aircraft state `x` and `w` and the control surfaces deflection demand `d` the inputs to fly properly the aircraft. These inputs fed to the

aircraft model must be obviously in accordance with the maximum allowed banking angle `M` and the maximum thrust (calculated from `s`).

It is not designed to be directly invoked by the user program, in fact it is usually invoked by `gsaml_boid`.

`void gsaml_rk(void (*fname)(double*,double*,double*),double *x, double *in, int n,double dt)`
Runge Kutta integration routine.

Integrate the function given as `(*fname)` over the time `dt`.

It is not designed to be directly invoked by the user program, in fact it is used by most of the other routines to perform all the integrations.

`void gsaml_kkkk(double Vt,double *kk)`
Variable feedback routine.

Compute the feedback gains according with the aircraft instantaneous speed `Vt`.

It is not designed to be directly invoked by the user program, in fact it is usually invoked by `gsaml_ics`

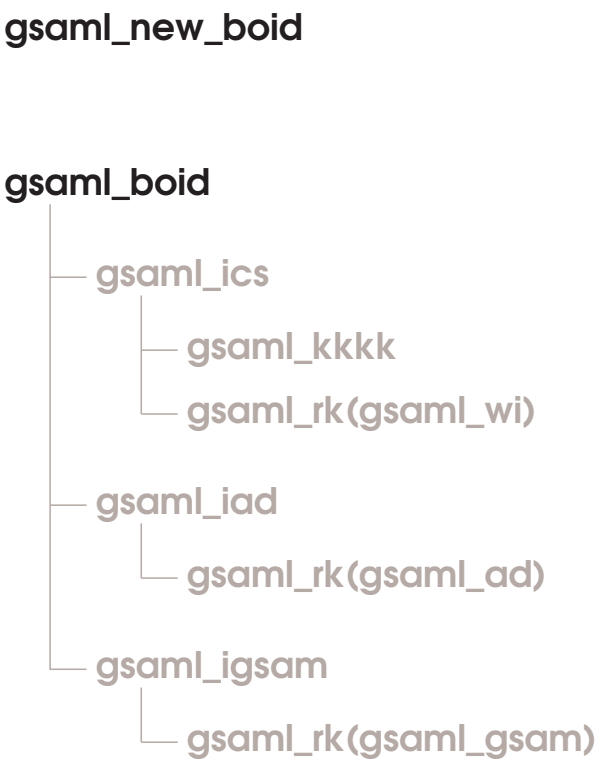

Figure 3.32: Hierarchical structure of the library. The form `gsaml_rk(function_name)` simply means that the function `gsaml_rk` is used to integrate the outcome of the `function_name` function.

Chapter 4

Flocking

4.1 Introduction

In this chapter a concrete application of the software routines developed in the two previous chapters is shown. This constitute an "on the road" validation of the capabilities of the aircraft model and control system.

Second aim of the simulation is also to demonstrate that a simple algorithm based only on visual information can be used to maintain a pre-existing flock during a random flight.

As last the influence of the limited view angle and view distance on the flocking capability are analysed throughout a set of simulations.

4.2 Flocking theory

4.2.1 Reynolds' model

The first and for some aspects the breakthrough work on the flocking theory, is the paper presented in 1987 by C. Reynolds [2].

In his work Reynolds presented a computer graphics animation of a bird flock in which the birds trajectory were not predefined, each boid was flying itself following a set of rules [1]. The animation resulting from this distributed algorithm looked similar to a natural flock and moreover was self-generated.

School of fish, flocks of birds and herds of land animals are all examples of aggregation developed by animals to improve their abilities; the Reynolds' model was directly inspired by considerations on these animals' aggregations.

Urges

Carefully analysing the behaviour of each member of a natural flock two basic urges can be individuated [19]:

- an urge to stay close to the flock (to the centre of its flock mates);
- a more obvious desire to avoid collisions with the flock mates.

[1] The flock simulated by Reynolds is closely related to particle systems, largely studied nowadays. Fundamental difference is that in a particle system, the sub-object are represented as dots they therefore do not have shape and a complex geometrical state as the boids.

Secondary needs as foraging for food or the migratory instinct can be also observed in animals; however these urges do not play an important role on the flocking mechanism.

In the Reynolds' model the two previous needs leads to a set of rules applied to fly the boids:

- *Collision Avoidance:* avoid collision with nearby flock mates;
- *Velocity Matching*: attempt to match velocity of the nearby flock mates;
- *Flock Centring:* attempt to stay close to nearby flock mates.

The rules are listed in order of decreasing priority; later on will be explained how the priority is used to carefully arbitrate them.

Collision avoidance try to fly the boids away from theirs flock mates when the static distance is shorter then a predefined threshold, independently from the boid speed. The velocity matching conversely works independently from the previous rule trying to match the boid's speed with the speed of its neighbours. In this way is realized a sort of predictive version of collision avoidance; in fact if two boids are flying with the same speed, they will probably not collide at least in the near future.

Flock centring drives the boid toward the centre of the flock. Must be underlined that the boid is able to see only a portion of the flock (its neighbours); therefore this rule will drive the boids towards this local center and not toward the center of the whole flock.

Capabilities

Stepping back to the birds nature, can be analysed the second class of factor that play an important role on the flocking; these are the fly and perception capabilities of the boid.

Fly

The aerodynamic of flight and the finite amount of energy utilisable by a bird, set strong limitation to its movements this therefore largely complicates the flocking. To account for this limitations, in this model is defined the geometric flight.

The geometric flight consider each boid as a rigid body, with its proper direction and orientation. The boid fly in the direction of its thrust, and the conservation of momentum is applied; a boid in flight tends to stay in flight. To emulate the boid's finite amount of power a simple viscous speed damping is defined; consequently a maximum speed became also defined. As normal for a natural creature also a maximum acceleration is defined directly as a percentage of the maximum speed.

To achieve a reasonable real turn behaviour, the concept of a coordinated level turn is applied. As seen in §3.4.3 during a level turn, the steering manouvre is in fact determined by the centripetal force, coming from the vector difference between the lift and the gravity force. As known lift is approximately proportional to the airspeed, while the path curvature is proportional to the turning force. All these dependence are accounted in the model considering a turning force

proportional to the boid speed and inversely proportional to the turn radius. The turn is simply realized changing the direction of the boid's thrust.

More realistic physical force and effect as gravity, loss of lift and the other effects discussed in cap.2 are in this model neglected.

Perception

During flocking real animals use theirs senses to acquire informations about the world and the neighbours. For birds, the sense principally involved in flight is clearly the vision. Rather then try to reproduce the birds' perception, the model filter-out the great amount of information available in the computer simulation to obtain the same information available for a real animal.

For this aim a sensor is defined, the sensor space is a spherical zone centred in the boid c.g., its sensitivity is defined as an inverse exponential of distance.

Reynolds' suggest in his model a sensitivity inversely proportional to the square of the distance because the resulting flocking looks more similar to a natural one. Moreover this suggestion reflects also another fact, the angle that an object occupies in the visual field of an observer, varies inversely with the square of its distance from the observer.

Needs to be underlined as the local perception, is one of the key points of this model. If in the flock centring rule the local centre of each boid is substituted with the global center, the result will be only a disordered motion with successive contraction and expansion.

From the way in which the algorithm is defined, is easy to understand that the capabilities needed by each bird, in order to flock, are mostly independent from the flock size. This idea is confirmed by the fact that in nature have not been actually observed upper bound to the flock size. The flocking operation of a single boid can be performed by a constant time algorithm, independently from the number of boids. Is clear therefore the high scalability of this approach.

Considering that a real bird has a field of view of approximately 300 degrees, and a stereo perception only in a small forward cone of 10-15 degrees, is suggest that a forward weighted perception would probably improve the model. In this way should be avoided the situation in which a boid is distracted by other boids in his back. Is also suggested that probably, the perception should be exaggerated in the forward direction by an amount proportional to the cruise speed,

Arbitrating the rules

The three different behavioural rules that must be combined together to determine the acceleration vector that drives the boid.

The simplest way to combine the request is obviously to average them in a vector sum. This approach works pretty well in normal cruise conditions but shows its limitation in critical conditions. If for example two different rules suggest two accelerations with opposite direction but more or less the same magnitude, the acceleration resulting from the sum will be approximately null. The boid will therefore practically ignore both the suggestions with the imaginable tragic consequence.

Exploiting the priorities defined along with the rules, better arbitrating mechanisms can be realized. A simple method suggested by Reynolds is to use a predefined amount of acceleration to fulfil the acceleration requests in order of

priority. In this way the most important urges will be listen as first avoiding tragic consequences; then if possible, will be followed also the other suggestions.

Migratory instinct

Using the model presented so far, a set of boids randomly placed in a finite space exhibits a flocking behaviour forming some flockettes or eventually maybe a unique flock. All these formation will swarm randomly in the simulation space without following any predefined path. To be able to exploit the benefit of the flocking in a concrete application, a way to define the boids path is needed.

A simple way to do this is to introduce an additional rules to the fundamental three, a rule defined as migratory instinct. The migratory instinct has to provide flying suggestion to point the boids towards a predefined target or onto a predefined path.

The migratory instinct is clearly less important then the first three rules, it thus has a lower priority.

4.2.2 Flocking of UAV

A second work presented by B. Crowther and X. Riviere [1] has some analogy with the present work, it deserve therefore to be introduced. This second work shows a possible flocking based solution for the automatic control of a large number of unmanned aircraft.

UAV model

Although it mostly applies the Reynolds' concepts, one major difference in this work is present.

The physic model of the boid is deeply changed; the boid is substituted with a 6 DOF mathematical model of a real aircraft, provided also with a control system. Conversely from the simplified boid model, this model is able to account for most of the physic phenomena which are present in a real flight. In particular, important phenomena as inertia, gravity and aerodynamic limitations, which will obviously effect the flocking capabilities, are correctly recreated.

As in a real aircraft the control system accept as input the desired speed heading and pitch and provides the correct control surface deflection to the aircraft model 3.

Perception

The boid perception is defined here in a different manner. The urge of each rule is expressed in the form of a suggested centroid. As first an imaginary spherical sensor of radius r_s, located ahead of the boid is defined. Then the neighbours present in the spherical region are accounted according to a Gaussian weighting function.

The inverse quadratic weighting function suggested by Reynolds from visual consideration is here abandoned for a more suitable and smooth inverse exponential function.

The weight associated to a neighbour at the distance d_{ij} from the sensor can be written as,

$$w_{ij} = e^{-\left(\frac{d_{ij}}{r_s}\right)^2} \tag{4.1}$$

where r_s is the sensor radius.

The centroid is then computed averaging the weight of all the near aircrafts;

$$\mathbf{X_{C_{Ri}}} = \frac{\sum_{j=1}^{n} w_{ij}\mathbf{e_{ij}}}{\sum_{j=1}^{n} w_{ij}}, \tag{4.2}$$

where $\mathbf{X_{C_{Ri}}}$ is clearly the vector centroid, while $\mathbf{e_{ij}}$ is the vector displacement between the sensor centre and the boid j.

The placement of the centroid sensor ahead of the boid's centre by a predefined distance has also to be noted. This reflect exactly the Reynolds's suggestion to use a forward weighted vision sensor to emulate the natural birds vision.

Rules

A slightly different set of rules is also used; in addition to the three rules already presented, a fourth rule defined alignment is used.

As understandable from its name, this rule take care of matching the heading of the near aircraft. The attractive or repulsive velocity suggestion for the rules presented are respectively defined as follow,

$$\mathbf{V_{D_R}} = W_R \frac{\mathbf{X_{E_R}}}{|\mathbf{X_{E_R}}|} \left(1 - e^{-\frac{\mathbf{X_{E_R}}}{r_R}^2}\right), \tag{4.3}$$

$$\mathbf{V_{D_R}} = W_R \frac{\mathbf{X_{E_R}}}{|\mathbf{X_{E_R}}|} \left(e^{-\frac{\mathbf{X_{E_R}}}{r_R}^2}\right), \tag{4.4}$$

where

$$\mathbf{X_{E_R}} = \mathbf{X_{C_R}} - \mathbf{X_B}, \tag{4.5}$$

W_R is the weighting factor and $\mathbf{X_B}$ is the boid centre.

The net velocity vector demand, rising from the behavioural rules is obtained simply through a weighted average (note W_R into 4.3 and 4.4) and no type of arbitration strategy are applied.

Despite of its simplicity, this strategy seems to work pretty well; however are still present situation of collision. The authors suggest that this situations could probably be solved using an evasion rule formulated as a very localised version of the avoidance rule.

Important outcome of the work is the demonstration that the capability of flocking is basically governed only by the cohesion and alignment rule.

In a second article [20] the same authors applying the geometric flight idea, showed that the ratio between the two weighted factor allow or deny the flocking. They also demonstrate how the changes of phase between different scheme of flocking are independent from the cruise speed of the aircraft.

4.2.3 Vision based zooids

Third precious source examined during the development of this work, is the paper about animal aggregation written by Garry D. Peterson and Arthur R. Marshall[21].

As declared by the authors, the aim of the work is discovering, from a theoretical ethological perspective, which realistic aggregation mechanisms can generate

realistic aggregate behaviour. In this paper the attention is principally pointed on a set of *vision-based* behavioural rules.

The simulated world of the zooids[2] is only two dimensional; while theirs motion is constrained by limits to theirs capabilities. A maximum and a minimum velocity are defined; also limits to the maximum (positive and negative) acceleration are set. Similarly the rate of change of theirs headings is also limited.

The novelty in this work is certainly the definitions of the zooid's perception. The field of vision is not omnidirectional as in the two previous work, but conversely it is limited to 180 degrees in the zooid's forward direction.

The definition of the sensor range is also based on the visual perception, a zooid can not recognise objects occupying a visual angle lower then 0.5 degrees. The visual angle (α) is defined as visible the angle that the object occupies in the zooid's visual field (see figure 4.1).

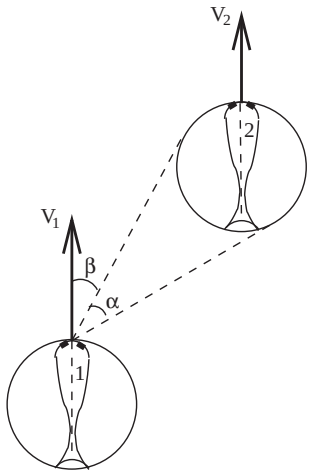

Figure 4.1: Zooids' visual angle

The behavioural rules are not specified in details but two important ideas are highlighted. The only two parameter involved in the rules are

$$Loom = \frac{d\alpha}{dt}, \tag{4.6}$$

$$\tau = \frac{2\alpha}{Loom}, \tag{4.7}$$

where *Loom* is the rate of change of α and τ is an optical estimate of the collision time. this choice was made to reflect the real amount of information effectively perceived by the zooids. No informations about direction or speed of the neighbours are considered as known, the rules are based only on visual clues. The speed information is substituted with an approximation obtained considering the α rate of change.

Being generally predefined the size of a zooid, the use of the visual angle as a perceived parameter instead of the usual distance embed automatically in the rule also a weighting function. The visual angle variation is in fact an inverse quadratic function of the distance as observed by Reynolds.

[2]Due to theirs 2D limitation the actor in this work are denominated Zooids following the precedent set by Reynolds' boids.

The zooid that occupies the largest portion of the zooid visual filed its considered the zooid's leader, the neighbourhood definition is in this model reduced to a single zooid.

A novel type of rules arbitration is also suggested; the proposed idea tries to simplify the decision making process. An hierarchical structure is presented; the final driving suggestion is simply the result of a set of true/false decisions. In each decision is involved only one of the perceived parameters.

In the simulation conducted using this model the zooids exhibits clear form of aggregation. The polarised group formed by the zooids, has the ability to aggregate if confined in a closed space.

4.3 Vision system

In this chapter is investigated the dependence of the flocking capabilities from the intrinsic limitations of the vision system. For this reason it is assumed that the only positional informations that an aircraft can exploit are the visual ones obtained by its inboard camera.

All the positional informations that can be retrieved from the on board camera are by definition relative; probably they are sufficient to mantain the flocking, however they are not enough to control all the flight operations (take off, landing, flock formation).

Although not considered here, each aircraft of the flock is also provided with other types of positioning systems (i.e. GPS)[3].

Absolute positional informations are used to perform navigation task (i.e. path following), and also to exploit specific task connected to the flocking. The aircrafts will not usually take off together, therefore due to the limited range of the camera, the vision flocking algorithm alone will not be able to join all the aircrafts in a single flock. The coarse informations of the absolute positional system will be then used to initially gather all the aircrafts together in a single flock.

The wide range of the absolute positional system will be useful also to make up the eventual failures pointing the lost aircraft back towards the flock. In this way the needed robustness for the flight controller can be guaranteed.

Along with specific systems, this second positioning system will be engaged during the take off and landing operation, as it is done currently by commercial GPS based autopilots for UAV.

4.3.1 Camera model

At this stage of the work is chosen to separate the study of the vision algorithm [4] and of the flocking algorithm. For this porpoise is defined a geometrical model of the vision system, which is then used to define whether a target aircraft is visible or not.

In figure 4.2, the 3D angular sector that represent the camera view field is shown. The camera is supposed as mounted in the aircraft centre of gravity and

[3] The standard resolution of a basic not expensive GPS set is generally only $\pm 25m$

[4] An ad hoc image processing algorithm is currently being developed by other members of the GRIDSWARM team.

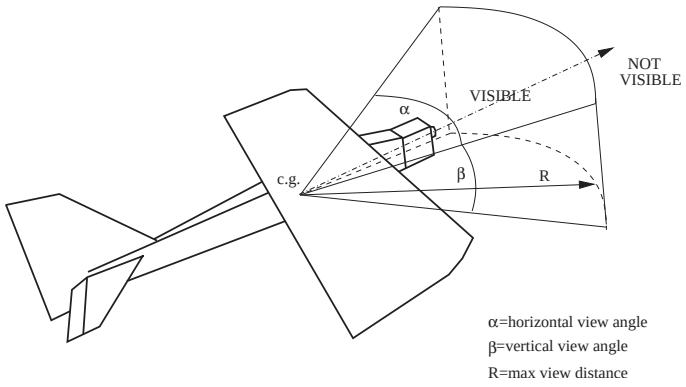

Figure 4.2: Model of the camera

its forward direction vector coincides with the aircraft body axis. The visible region is completely defined by the vertical view angle α, the horizontal view angle β and the maximum view distance d.

For simplicity each aircraft is considered as a single point which coincides with its c.g., therefore an aircraft is defined visible when its c.g. is inside the vision region.

Within the simulation program, all the state variables of each aircraft are obviously well known; conversely with the images analysis only some informations can be retrieved. The flocking algorithm must be then totally based only on this small amount of informations.

Obviously the informations retrieved and theirs resolution, depends on the processing executed on the source images. However referring to the general capabilities of the computer vision programs diffused nowadays, it can be assumed that the vision algorithm is able to provide an estimation of the angles $\tilde{\alpha}$ and $\tilde{\beta}$ and of the target distance $\tilde{D}$ see figure 4.3.

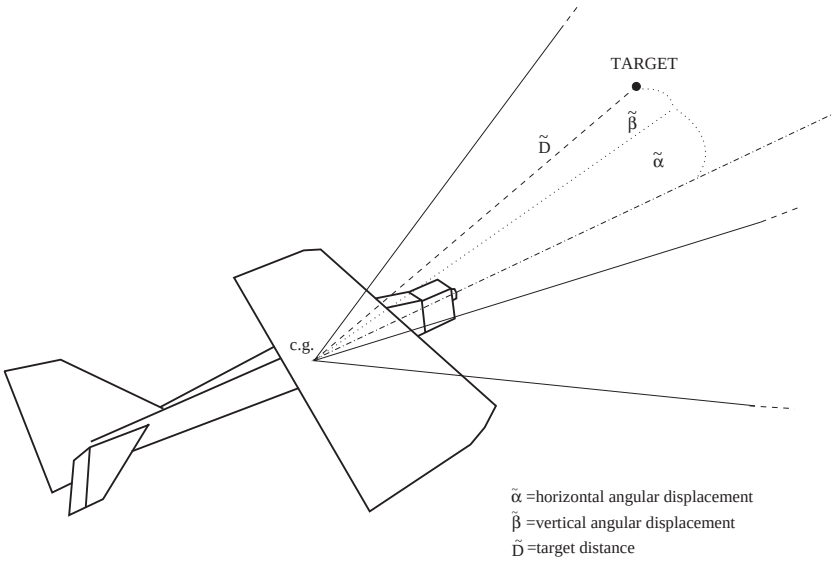

Figure 4.3: Parameter assumed as retrieved by the vision algorithm ($\tilde{\alpha}, \tilde{\beta}, \tilde{D}$ of the camera

All the flocking rules set is thus based only an those three simple informations.

4.3.2 Autopilot control inputs

After consideration about the input informations it is time to consider also the manouvre that must be suggested to the aircraft control system to perform the flocking.

Since the vision is the only input, great importance is given to the ability of keeping the target point inside the view field.

The best way to achieve this aim is to fly the aircraft in a way that step by step tries to reduce the angular displacements $\tilde{\alpha}$ and $\tilde{\beta}$ between the aircraft forward direction and the target. The role of the flocking algorithm is to compute the required $\tilde{\alpha}$ and $\tilde{\beta}$ angular displacement, while the control system has to fly the aircraft controlling the aircraft flying service in order to obtain the required manoeuvres.

The decision of designing specifically a pitch control autopilot and a jaw control autopilot, is driven by the necessity to have the best possible control on $\tilde{\alpha}$ and $\tilde{\beta}$. It is instructive to highlight that this is not exactly the same approach used commonly on an autopilot. Usually when an autopilot is used to follow a predefined flying path, two of its control subsystems are engaged; the altitude hold autopilot and the heading hold autopilot. The aim of the heading hold autopilot is to reduce the lateral angular displacement between the aircraft forward direction and the flight path. The aim of the altitude hold is to perform in a predefined time the manouvre necessary to keep the desired altitude. The elevator input provided by the altitude hold system is proportional to the altitude difference, but almost unrelated to the real angular displacement $\tilde{\alpha}$. The resulting manouvre can then lead to loose the target point.

For example, if the target is flying far away at an altitude slightly higher then the pursuer aircraft, the correct manouvre to increase slowly the pitch by an amount proportional to $\tilde{\alpha}$, when the desired altitude is reached, $\tilde{\alpha}$ will be null, therefore the aircraft will proceed in level flight. An altitude based autopilot conversely will rapidly increase the pitch by a large amount to ensure that the desired altitude is gained quickly; then it will stabilize back in level flight.

Maybe the result seems the same but it is not, in fact the large inclination initially gained has probably got the intolerable effect of loosing the target.

4.4 Flocking algorithm

The flocking algorithm proposed here is directly inspired by the works presented in §4.2, however the use of a limited vision imposes some changes.

It is worthwhile to remember that it is assumed that the estimation of the angles $\tilde{\alpha}$,$\tilde{\beta}$ and of the target distance $\tilde{D}$ are assumed as the only information that the vision algorithm is able to provide.

Rules

The main rules applied to obtain the flocking are only two:

- *avoidance rule*: urge to avoid collision with the flock mates (see fig.4.4);

- *cohesion rule*: urge to be within the near flock mates (see fig.4.5).

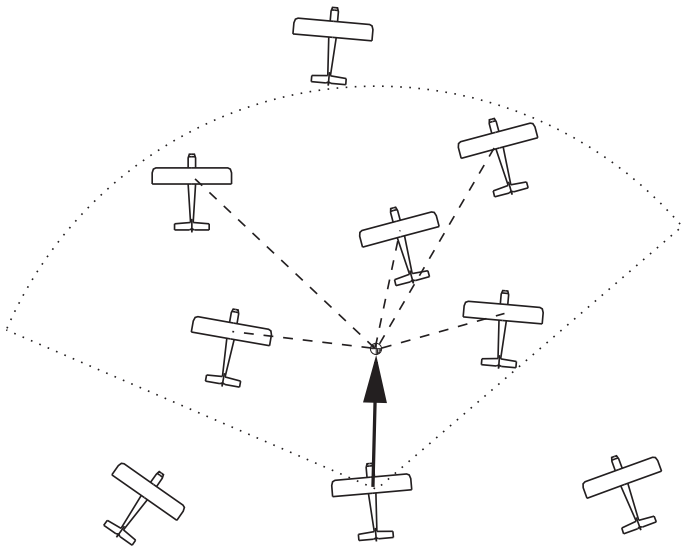

Figure 4.4: Cohesion rule

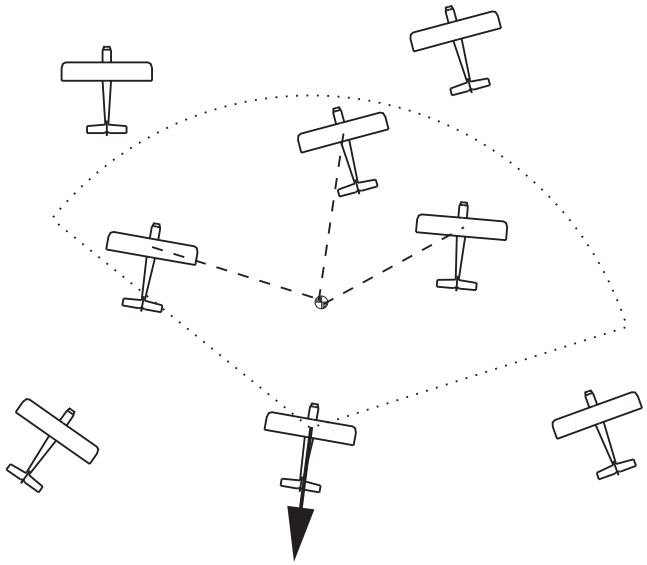

Figure 4.5: Avoidance rule

A *velocity regulation* rule is then obtained exploiting distance informations, however it will be shown that it does not properly correspond to the common formulation of the velocity matching rule.

The velocity matching rule or the orientation rule in the form proposed respectively in [2] and [1] are under the hypothesis of this work not directly applicable. In fact speed or heading of the neighbours aircraft must be considered totally unknown because they can not be directly retrieved using only a vision algorithm.

The contribute of each rule is initially formulated as a simple force; then the two forces are combined to obtain the total driving force. From this one, finally the flying suggestion are computed.

Centroids

As proposed in [1] for each one of the rules a correspondent centroid is computed.

The camera model presented in §4.3 defines the sensor view field. The positions of the aircrafts included into the sensor range are averaged accordingly to a predefined weighting function. The assigned weight is an inverse function of the distance in order to impose a stronger influence coming from the closer neighbours.

Several weighting schemes were tested within the algorithm development; and particular attention was dedicated to the inverse quadratic an inverse cubic function because directly inspired by vision consideration. No great difference were noted, however the Gaussian weighting function proposed on [1] was preferred because able to give slightly better results.

The weight associated by the sensor of the aircraft i to an aircraft j can be written as,

$$w_{ij} = e^{-\left(\frac{d_{ij}}{r_s}\right)^2} \tag{4.8}$$

where r_x is the sensor parameter and d_{ij} is the distance from the sensor origin.

The centroid is then computed averaging the weight of all the aircrafts ($j = 1..n$) into the sensor space;

$$\mathbf{X_x} = \frac{\sum_{j=1}^{n} w_{ij}\mathbf{e_{ij}}}{\sum_{j=1}^{n} w_{ij}} \tag{4.9}$$

where $\mathbf{X_x}$ is the vector centroid of the rule x, while $\mathbf{e_{ij}}$ is the distance vector between the sensor centre and the boid j.

Two different sensors (one for each rule) are considered; both of them cover a 3D angular region and both of them have the same horizontal and vertical view angles α and β. However they do not have the same maximum view distance and the same sensor parameter in theirs weighting functions.

Generally the range of the avoidance rule sensor is shorter then the range of the cohesion rule sensor. The shorter range of the avoidance sensor allow to consider only the closer neighbours which are the only that in the near future can lead to problems of collision. Also the sensor parameter of the avoidance rule is of course lower, to have the full excursion of the Gaussian function within the sensor range.

The maximum view distance as commonly intended, is actually then the range of the cohesion sensor; all the aircraft far away more than this distance can not be seen. The flocking capabilities are thus strongly related to this parameter.

In figure 4.6 and 4.7 are shown for clarity the plots of the two weighting function as function of the distance from the sensor centre.

Forces

The centroids are now defined; an expression for the forces towards the centroids has to be defined.

The cohesion rule exercise a positive force towards the cohesion centroid and the magnitude of the force must be proportional to the distance from the centroid. The expression of the force is defined by a Gaussian function of the distance,

$$\mathbf{F_c} = \frac{\mathbf{X_c}}{|\mathbf{X_c}|}\left(1 - e^{-\left(\frac{\mathbf{X_c}}{r_c}\right)^2}\right). \tag{4.10}$$

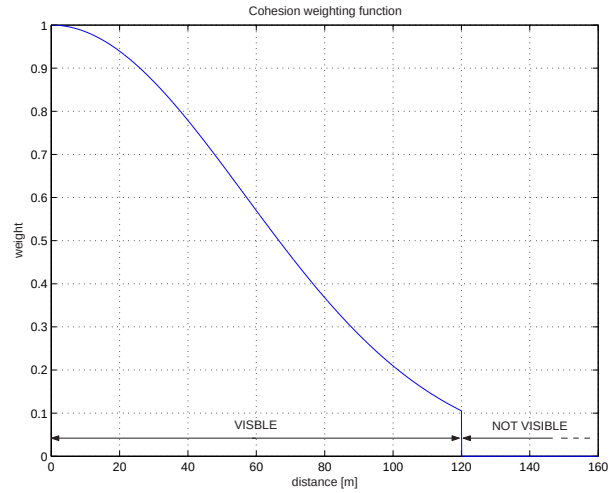

Figure 4.6: Cohesion weighting function

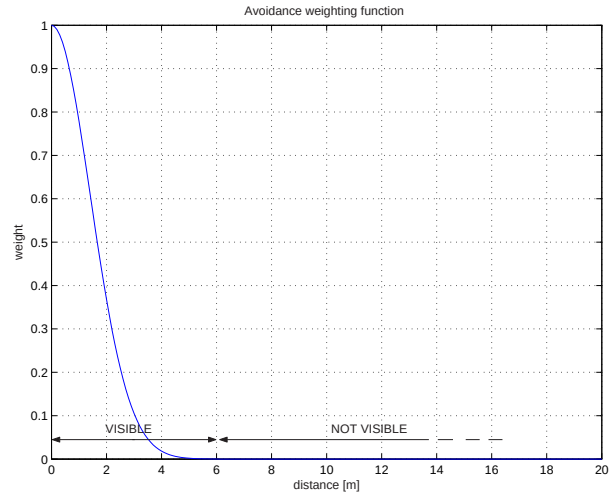

Figure 4.7: Separation weighting function

where r_c is the sensor parameter, and $\mathbf{X_c}$ is the vector displacement between the sensor centre and the centroid. In figure 4.8 the magnitude of the cohesion force is plotted as function of the distance from the centroid.

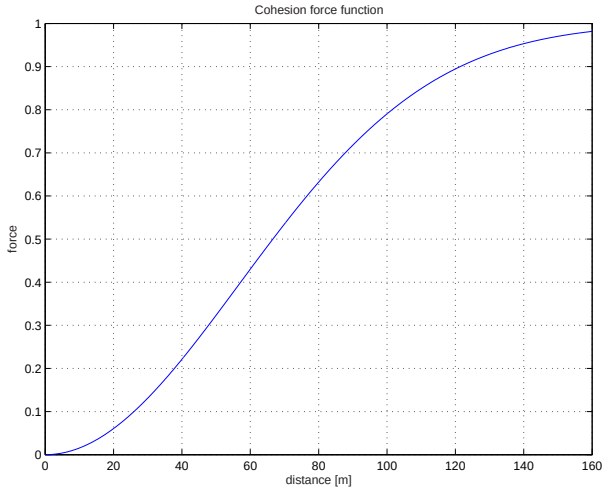

Figure 4.8: Cohesion force vs distance from the centroid

Vice versa the avoidance rule exercise a negative force towards its centroid and the magnitude of the force has to be inversely proportional to the distance from the centroid. The expression of the force is defined by a negative Gaussian function of the distance,

$$\mathbf{F_a} = -\frac{\mathbf{X_a}}{|\mathbf{X_a}|}\left(e^{-\left(\frac{\mathbf{X_a}}{r_a}\right)^2}\right), \tag{4.11}$$

where r_a is the sensor parameter, and $\mathbf{X_a}$ is the vector displacement between the sensor centre and the centroid. In figure 4.9 the magnitude of the avoidance force is plotted as function of the distance from the centroid.

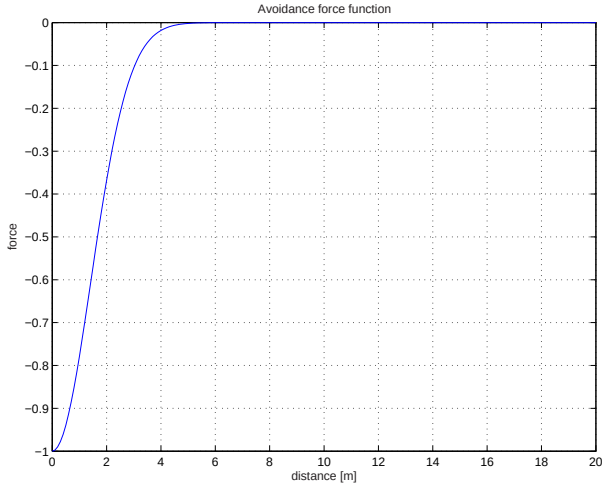

Figure 4.9: Avoidance force vs distance from the centroid

Arbitrating the forces

A weighted vector sum is chosen as method to combine the attractive and repulsive force suggested by the flocking rules.

The final driving force can therefore be calculated as:

$$\mathbf{F_d} = w_c\mathbf{F_c} + w_a\mathbf{F_a}. \tag{4.12}$$

The two weighting factor has to be carefully chosen. The ratio between the two factor determines the prevalence of one of the two rules, while the absolute value of each weight determine the maximum value of the respective force.

In particular large value of the avoidance weighting factor leads to oscillation of the formation and must therefore be avoided. During all the simulation presented here the values $w_c = 1$ and $w_a = 0.5$ were used. To clarify the joint action of the two forces, in figure 4.10 the magnitude of the total force is plotted as function of the distance. Please note that the situation in which the cohesion and avoidance centroids are coincident is supposed.

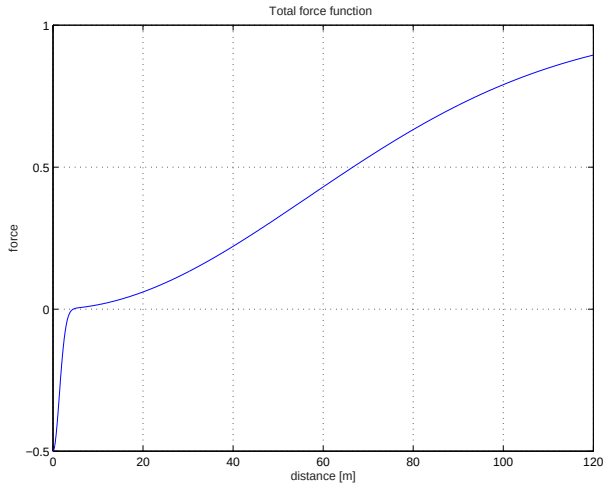

Figure 4.10: Total force vs distance when the avoidance and cohesion centroids are coincident

Other type of dependence from the distance were also tested for the forces, these were inverse proportional, inverse quadratic and inverse cubic function of the distance. Although they seem to be suitable at large distance, they present an heavy drawback when the distance is short. At short distance the magnitude of the force can be extremely high; in this situation the action undertaken by the navigation module is quite unpredictable.

During the simulations strange behaviour were very frequent. For example an aircraft initially approaching its neighbours, was then violently repelled when slightly closer. The action was usually so violent to fly away the aircraft from the flock.

The Gaussian function is for this reason more suitable, it is in fact particularly smooth at large distance and also limited for the small one.

From forces to guidance inputs

Obtained the direction and the magnitude of the vector acceleration suggested to perform the flocking, it is decomposed to obtain the guidance inputs.

From the components along the forward axis ($\mathbf{F_n}$) and the lateral axis ($\mathbf{F_e}$) is computed the suggested jaw angle,

$$jaw = \arctan\left(\frac{\mathbf{F_e}}{\mathbf{F_n}}\right). \tag{4.13}$$

Similarly is computed the suggested pitch,

$$pitch = \arctan\left(\frac{\mathbf{F_z}}{\mathbf{F_n}}\right). \tag{4.14}$$

Limitations are imposed to the *pitch* in order to avoid undesired stall situations. Conversely no limitation are imposed to the suggested *jaw*; must be remembered that into the lateral autopilot is already present a limitation to the maximum roll angle, which avoid anomalous behaviours.

Finally the speed variation from the cruise speed is adjusted proportionally to the forward component of the total force ($\mathbf{F_n}$). A positive value of the forward force will therefore increase the aircraft speed, while a backward oriented force will reduce it. The maximum variation from the cruise speed is proportional to the 20% of the cruise speed.

4.5 Simulations

To understand up to what extent the vision limitations can influence the flocking, a complete set of simulations was run. The investigation cover the effect of the variation of the two fundamental parameters which characterise the vision; the maximum view distance and the horizontal view angle. In all the simulations exactly the same flocking algorithm (the one presented in the previous paragraph) is considered; also the vertical view angle is fixed to 80 degrees.

This limitation obviously reduces the aircraft awareness, however due to the simplicity of the flocking algorithm this is strongly needed. The flocking algorithm developed so far, works quite well in the horizontal plane. It is in fact able to perform more or less the same manoeuvres that a human pilot would do to follow a target. When not too close to its target, it works pretty well also in the vertical plane correcting the aircraft pitch to match the target altitude. However in a situation in which the target is directly above or under itself, this algorithm leads to silly manoeuvres.

In those situations the vector decomposition of the total force suggest violent pitch down or pitch up manoeuvres which can jeopardise the aircraft stability. Due to the fact that the final aim is to achieve a planar formation, this limitation will not have considerable impact on the final result of the work.

The magnitude of 80 degrees for the vertical view angle was decided after a series of tests. This magnitude guarantees to avoid wrong manoeuvres, while at the same time insure the necessary awareness to match the target altitude.

Scenario

The types of simulations performed can be divided in two different types; a first qualitative simulation to prove the flocking ability, and a second group of quantitative simulation.

In the first simulation the aircrafts are placed in random position and with random orientation within a closed square space. The only flocking rules engaged are the cohesion and the separation rule. If the aircraft is not able to see any of the mates, it will simply maintain the steady state flight. If an aircraft which is not following one of its mates, cross one of the boundaries, the flocking algorithm will suggest a turn manouvre to drive it back into the simulation space. At the end of the time history simulation, the aircrafts trajectories are evaluated qualitatively.

In the second group of simulations, to study the flocking, a typical scenario of use was set up. To avoid the already explained problem of flocking formation, the aircraft are started from a predefined triangular shape planar formation (see fig.4.11). The distance between the flock members are defined according to the

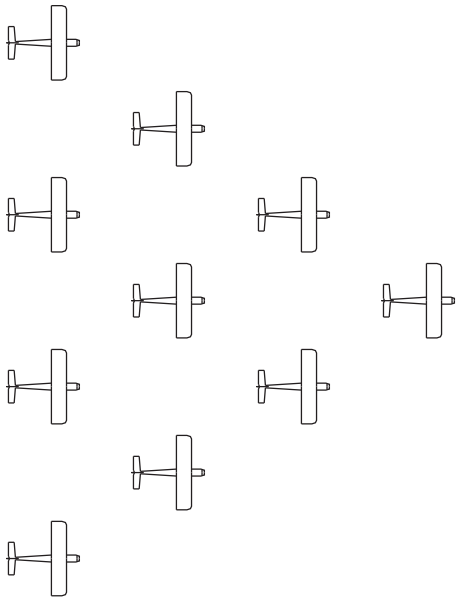

Figure 4.11: Initial triangular shape formation

camera view angle; this ensures that in the starting formation each aircraft (a part from the first) is able to see at least one of its neighbours. In the case of narrow view angle the in line formation (see figure 4.12) is preferred to the triangular one because it would require to place the aircrafts too close to each others.

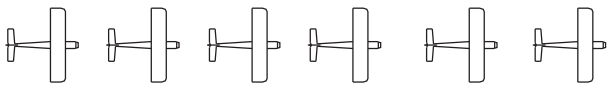

Figure 4.12: Initial in line formation

The flocking algorithm is designed to provide the aircraft with the flying suggestion. However if no neighbours are visible, the migratory instinct rule

should be used to determine the flying suggestion, creating in this way a sort of navigation module for the aircraft. Probably also for the gridswarm flock this will be the strategy to define a target or a predefined flight path.

To conduct tests on the vision system, the use of a pre-defined flight path is not a brilliant choice, the results will in fact dependent from it and therefore they will be not so significant. A random pre-generated flight path probably seems a simple and correct solution, but it is not. In all the cases in which is used a flight path the aircrafts are not aggregated only by the action flocking algorithm but also by the fact that they are heading to the same control point of the flight path.

A completely different strategy is used; when an aircraft is not able to see other mates its flocking algorithm generates a complete random bearing manouvre. During the in flock flight, the only aircraft which is almost always not able to see other flockmates is the leader of the flock; generally then as result of this strategy, it will be flying randomly while the other aircrafts will try to follow it. If during a manouvre one of the aircrafts loose the flock, it will not be driven towards the flock in any explicit or implicit manner. The random manouvre generated by its flocking algorithm could lead it to rejoin the flock as well as flying it away. The flocking result are in this way, completely due only to the two flocking rules.

Due to its inertia, during a time step the aircraft will only start a suggested manouvre, therefore if a new random manouvre is generated each time step, the result will be a series of small left and right rolling manoeuvres. To leave to the aircraft the time to perform the suggested manoeuvre and then eventually to stabilize after a manoeuvre no more manoeuvres should be generated for a while.

Following these ideas, the magnitude of the bearing manouvre is generated according to a uniform random distribution between $-\pi$ and π, while the time between two manoeuvres is generated according to a geometric distribution.

$$p(k) = q(1-q)^k, \tag{4.15}$$

which has mean equal to

$$m = \frac{1-q}{q}. \tag{4.16}$$

The mean time is set equal to the time needed by an aircraft to perform a 360 degrees which is approximately 200 time steps (10 seconds). The value chosen for q is then 0.005.

Along with the random bearing the control system is fed also with the pitch and the speed input necessary to maintain a stable flight.

Flow diagram

The flow diagram of the program realized for the time history simulations, is visible in fig.4.13. The aircrafts are first of all initialised; depending on the type of simulation, the P_n, P_e coordinates and the heading are chosen randomly or to match a triangular formation. All the other state variables are initialised to a steady state value. The appropriate values for the steady state variables are selected according to the cruise speed.

The main loop constitutes the rest of the program. At each iteration, the flocking manoeuvres are calculated from the aircraft position, and the aircraft

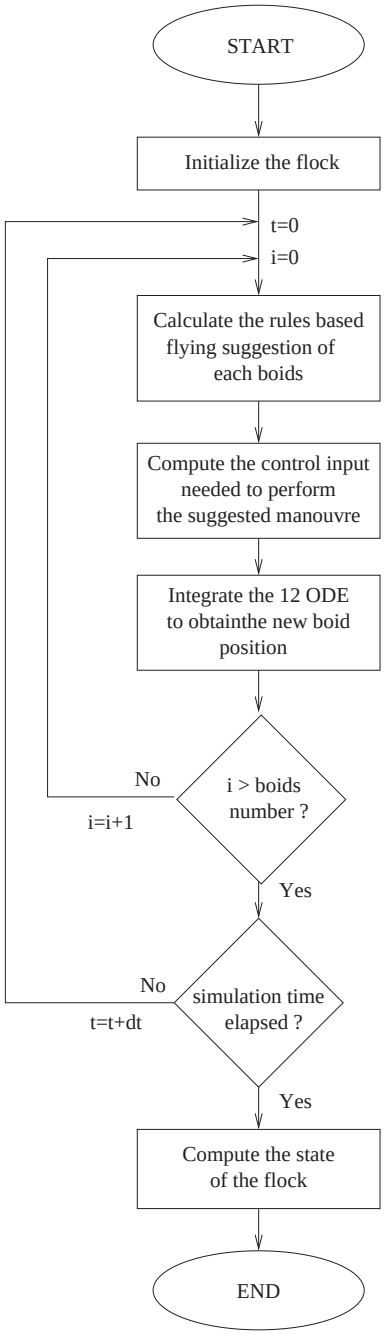

Figure 4.13: Flow diagram of the simulation program

model input are computed. The differential equations that constitute the aircraft model are integrated to obtain the state variables at the time $t + \delta t$. When the predefined time expires, the global parameters of the flock are calculated.

For obvious motivation of space all the code is not reported in appendix, it will be downloadable from the gridswarm website.

4.6 Results

Aggregation capability

The first simulation points to show the aggregation capabilities of the flocking algorithm.

As explained in the previous paragraph, for this simulation 20 aircrafts are started randomly in a closed square space of 1000 by 1000 meters. The aircraft will then fly freely, driven by the flocking rules but avoiding the boundaries. Although not so important for a qualitative evaluation of the result, must be specified that the other parameters of the simulation were fixed to typical values. The cruise speed was set to $15\frac{m}{s}$, the maximum view angle to 240 degrees the maximum view distance to 120 m and the total simulation time was of 50s. The final result of the simulation is shown in figure 4.14.

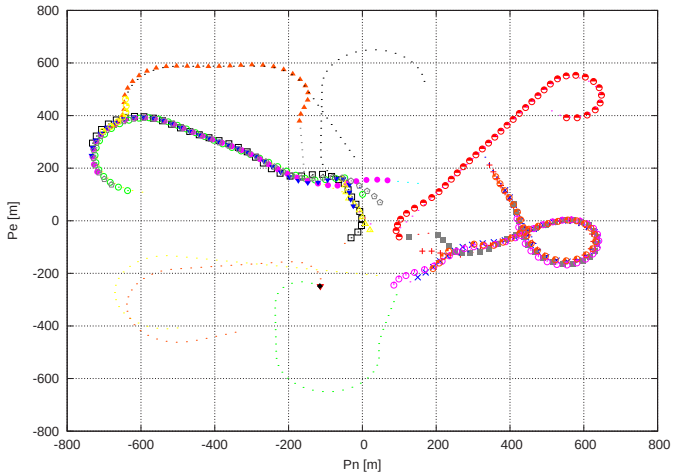

Figure 4.14: Aircraft trajectory after 50 *s of simulation. The trajectories of aircraft flying alone are indicated with dots, the remaining symbols indicate aircrafts flying in formation. The space between each aircraft symbols is* 1.25*s*

As expected despite of the limited vision the aircrafts show capabilities of aggregation. Even when the presence of the boundaries cause strong variation of the aircraft direction, the aircrafts have the capability to maintain the flock formation; this clearly suggest positive results for the following simulations.

Running several times the simulation was noticed that the aircraft tend to aggregate into an in-line formation, despite of the fact that they are started in a triangular one. Although other aggregation structures appear, the in-line one seems to be the most stable.

Flock keeping

The second simulation aims to understand how the view angle and the maximum view distance determine the flocking capability.

As already said, 10 aircrafts are started in formation and the simulation is run for the predefined number of time step, which was fixed to 2000 (correspondent to $100s$). The chosen duration time allows to have a complete reorganisation of the flock, the results are thus not dependent from the initial formation.

When the simulation is concluded, the state of the flock is checked; if any aircraft of the flock can see or is seen by at least one mates, the flock is considered intact.

Technically this check is done considering the flock as a graph in which each aircraft is represented by a node. The arcs of the graph represent the neighbourhood relation between mates. Checking if the graph is connected correspond then to check if the aircrafts still form one single flock.

Averaging the results of a series of simulations, the mean probability to have all the aircraft in a single flock after the simulation time can be computed. Being the input guidance (to the aircrafts not able to see other mates), completely casual, this probability is proposed as index of the flocking capabilities.

The number of iterations averaged to calculate the probability was fixed to 100, mainly to limit the time took by the simulation. However the probability values showed a substantial stabilisation when averaged on a number of simulations larger then 80. The cruise speed of the aircraft was set to $18\frac{m}{s}$.

The simulation was repeated several times varying the view angle and the view distance. The angle was varied from 10 to 360 degrees with a step of 10degrees, while the distance was varied from 40 to $120m$ with a $8m$ step.

The values obtained are plotted in figure 4.15.

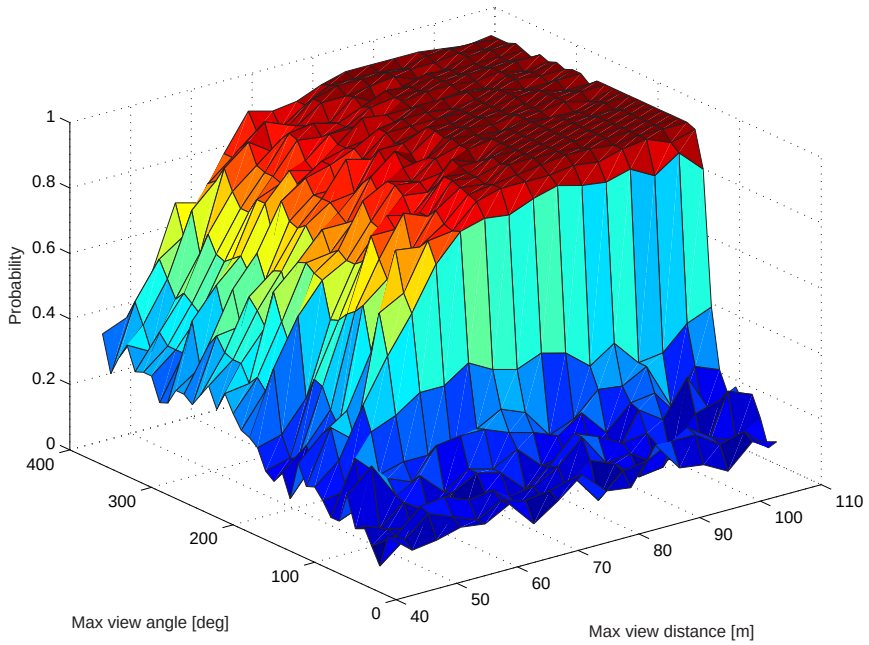

Figure 4.15: Mean flock keeping probability as function of the view angle and of the maximum view distance (3D plot). $V_t = 18\frac{m}{s}$ $t_{sim} = 100s$

A first look to the plot reveals immediately that large value of the view angle and of the view distance ensure better flocking capability. This evidence match the simple idea that the more is the awareness, the more effective can be the flocking.

Notifiable is also the fact that with the proper distance and view angle the flock is able to *maintain its cohesion in more then the 95% of the times.*

Redrawing the same results in the form of a coloured surface (see figure 4.16), some other interesting considerations can be made.

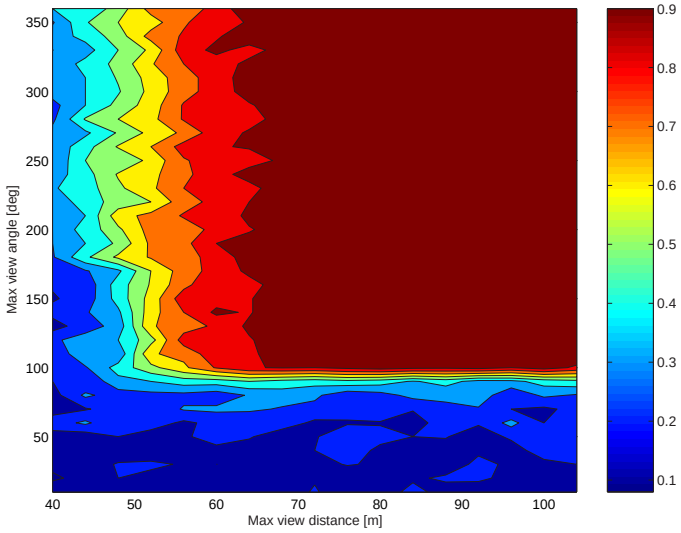

Figure 4.16: Mean flock keeping probability as function of the view angle and of the maximum view distance (colour surface plot). $V_t = 18\frac{m}{s}$ $t_{sim} = 100s$

The first impression is that the mean probability seems to be *depending separately from the view angle and from the maximum view distance.* Within the approximation due to the fact that simulated data are used; the border of the isoprobability surface are in fact almost parallel to the x and y axes. A second interesting consideration can be made observing the lower right part of the chart. In this zone the border of several isoprobability zone are parallel and very close. The probability, initially low for small view angles, seems to rise rapidly in correspondence with a defined view angle.

From a two-dimensional plot (figure 4.17) of the probability as function of the view angle, appears clearly that *exists a defined value of the view angle that guarantees to achieve the maximum flocking capability.* Excepted the cases in which the distance is lower than a minimum value (note the green plot corresponding to a max view distance of $52m$).

If the view angle is larger than *110 degrees*, the full flocking capabilities are achieved *independently from the distance* (note the red and blue lines corresponding respectively to a max view distance of 76 and $96m$). For clarity not all the inspected distances are reported in the graph however, above the needed minimum distance the curves are almost coincident.

To better understand the max view distance contribute, a plot of the probability as function of the view distance is reported in figure 4.17. Also in this situation not all the curves are reported. Three different curves one for each

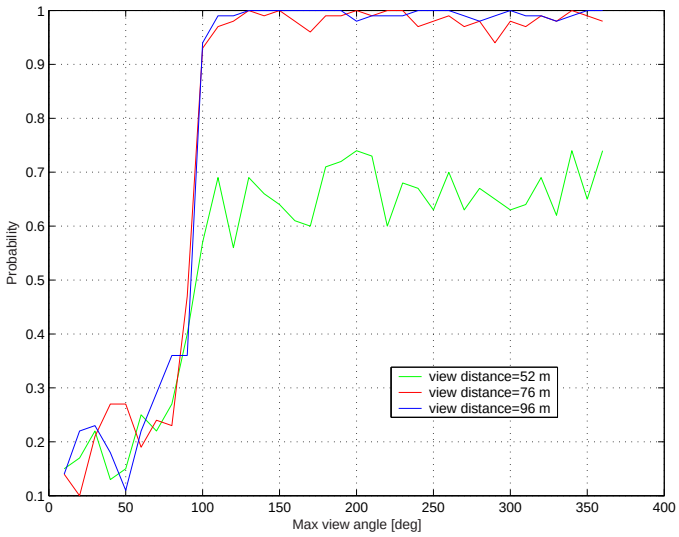

Figure 4.17: Mean flock keeping probability as function of the view angle. $V_t = 18\frac{m}{s}$ $t_{sim} = 100s$

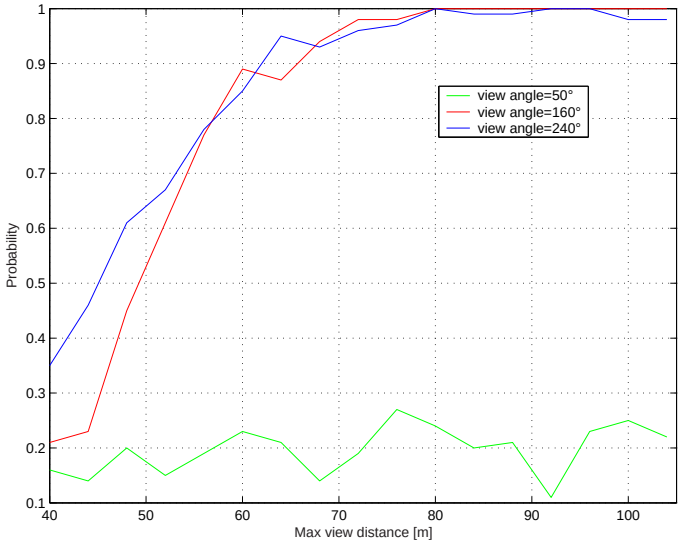

Figure 4.18: Mean flock keeping probability as function of the maximum view distance. $V_t = 18\frac{m}{s}$ $t_{sim} = 100s$

different behaviour are chosen.

For angles smaller then the threshold angle of 110 degrees (note the green line), as already underlined the flocking capability is very poor. Above the threshold view angle, *the flocking capability depends directly on the maximum view distance.* Independently from the view angle (note the red and blue lines), for distances larger than $80m$ the behaviour is almost the same; differences can be noticed for shorter distances. In particular the red line is indicative of the curves for view angle lower than 180degrees, and exhibit poor performance for low distances than its counterpart blue, which is indicative of the curves for view angle higher than 180 degrees.

Speculation on these result can be made thinking that, with a view angle larger than 180 degrees, also the in-row formation (see figure 4.19) is a type of

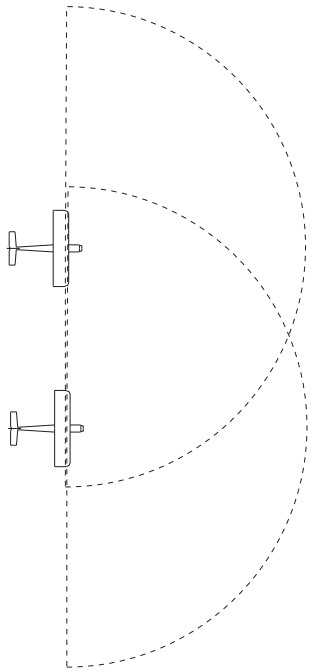

Figure 4.19: Stability of an in-row formation with a view angle larger than 180 degrees.

stable formation. This additional possibility increase slightly the flocking chances with low view distances, however it does not make any difference when the view distance is sufficiently high.

The obtained results prove that a vision based flocking algorithm can ensure the cohesion in more than the 95% of the times. The minimum view distance (80 m) and the minimum view angle (110 degrees) are not impossible requirements; they can be easily met by a camera with a proper lens or by a set of two cameras.

As suggested from the initial speculations, *the vision based approach is definitely suitable for the flocking.*

Conclusion

This thesis remarks that the dynamic of an aircraft can be fairly simulated throughout a simple software model. As well the aircraft can be controlled by a quite simple autopilot. The first aim of the work is achieved implementing both the systems in a single and easy to use software library.

Is confirmed that aggregated motion can be obtained using the simple rules of aggregation and separation also in the case of limited vision. Moreover is demonstrated that independently from the flight pat followed, with this simple two rules the cohesion of a preformed flock can be maintained with high probability. The only two factor which determine this capability are the max view angle and the max view distance. The simulations proved that a view angle larger than 110 degrees and a maximum view distance larger then 80 meters, are sufficient to ensure the flock coesion in more than the 95% of the times.

The tendency of creating an in-line formation showed by the algorithm, suggest that some other visual clues should be exploited to improve the flocking. A close integration with the vision algorithm is suggested, this will probably allow also a better strategy for the rules arbitration.

Appendix A

Symbols parameters and matrices

To help the reader all the symbols used are recalled in this appendix. An effort was made to use the symbols commonly used in the aerodynamic field.

A.1 Model parameters

In this section all the coefficients and parameters of the *GSAM* model are collected for an easy consultation.

SYMBOL	VALUE	DESCRIPTION
g	9.81	gravity acceleration $[ms^2]$
ρ	1.225	air density supposed constant$[\frac{kg}{m^3}]$

Table A.1: Environmental parameters

SYMBOL	VALUE	DESCRIPTION
$\bar{s}$	0.365	wing area $[m^2]$
$\bar{b}$	1.46	wing span $[m]$
$\bar{c}$	0.25	wing mean chord $[m]$
$d_{c.g.}$	0.25	CG position $[m]$
a_{wi}	0.035	wing incidence $[rad]$
m	2.3	mass $[Kg]$
I_{xx}	0.6	moment of inertia (X) $[Kgm^2]$
I_{yy}	0.11	moment of inertia (Y) $[Kgm^2]$
I_{zz}	0.30	moment of inertia (Z) $[Kgm^2]$
I_{xz}	0	moment of inertia (X/Z) $[Kgm^2]$

Table A.2: Geometrical and inertial parameters

A.2 Matrices

In this section all the matrices used in this work are reported in detail. The derivation of the reported matrices is omitted, nevertheless complete derivation can be found in [16].

SYMBOL	VALUE	DESCRIPTION
α_{ML}	0.297	alpha at maximum lift $[rad]$
C_{L_0}	0	lift at zero alpha
C_{L_α}	4.64	lift curve slope per radian $[\frac{1}{rad}]$
$C_{L_{tail}}$	0.40	coefficient of tail lift
C_{D_0}	0.038	profile drag
$C_{D_{CL}}$	0.065	induced drag
C_{S_β}	-0.52	sideforce from sideslip
$C_{S_{\delta r}}$	0.186	sideforce from rudder
C_{m_0}	0.072	pitching moment at zero alpha
C_{m_α}	-0.72	pitching moment with alpha
$C_{m_{\delta e}}$	-1.12	pitching moment with elevator
C_{m_Q}	-9.07	pitch rate damping
$C_{m_{\dot{\alpha}}}$	-3.63	pitch damping with alpha rate
$C_{l_{\delta r}}$	0.01	rolling moment with rudder
$C_{l_{\delta a}}$	-0.35	rolling moment with aileron
C_{l_P}	-0.45	roll rate damping
$C_{l_{\beta 0}}$	-0.02	rolling moment with beta
$C_{l_{\beta_{C_L}}}$	-0.05	rolling moment with beta variation due to C_L
$C_{l_{R0}}$	0.0112	roll rate damping
$C_{l_{R_{C_L}}}$	0.16	roll rate damping variation due to C_L
C_{n_β}	0.05	yawing moment with beta
$C_{n_{\dot{\beta}}}$	0.0	yawing moment with beta rate
$C_{n_{\delta r}}$	-0.04	yawing moment with rudder
$C_{n_{\delta a}}$	0	yawing moment with aileron
C_{n_P}	0.013	yaw rate damping
$C_{n_{R0}}$	-0.07	yaw rate damping
$C_{n_{R_{C_L}}}$	-0.0175	yaw rate damping variation due to C_L

Table A.3: Aerodynamic coefficients

$$\mathcal{C}_{BE} = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & \sin\phi\cos\theta \\ \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi & \cos\phi\cos\theta \end{bmatrix}$$

$$\mathcal{C}_{BW} = \begin{bmatrix} \cos\alpha\cos\beta & -\cos\alpha\sin\beta & -\sin\alpha \\ \sin\beta & \cos\beta & 0 \\ \sin\alpha\cos\beta & -\sin\alpha\sin\beta & \cos\alpha \end{bmatrix}$$

$$\mathcal{C}^*_{EB} = \begin{bmatrix} 1 & \tan\theta\sin\phi & \tan\theta\cos\phi \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix}$$

$$\mathcal{C}_{SB} = \mathcal{C}_{SB}^{*} = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}$$

$$\mathcal{C}_{BS} = \mathcal{C}_{SB}^{T} = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix}$$

$$\mathcal{C}_{d_{c.g.}} = \begin{bmatrix} 0 & 0 & 0 \\ \bar{c}(d_{c.g.} - 0.25)\sin\alpha & 0 & \bar{c}(d_{c.g.} - 0.25)\sin\alpha \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathcal{C}_{w} = \begin{bmatrix} 0 & -R & Q \\ R & 0 & -P \\ -Q & P & 0 \end{bmatrix}$$

$$\mathcal{I} = \begin{bmatrix} I_{XX} & 0 & -I_{XZ} \\ 0 & -I_{YY} & 0 \\ I_{XZ} & 0 & -I_{ZZ} \end{bmatrix}$$

Appendix B

Software programs

In this second appendix, part of the software programs used in this work are listed; the rest will be downloadable from the gridswarm website.

B.1 GSAM model

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                            GSAM core model                                   %
%                                                                              %
% INPUT:                                                                       %
% x=[Vt,a,b,theta,phi,psi,Q,P,R,Pn,Pe,Pz]                                      %
% u=[de,da,dr,thrust]                                                          %
% OUTPUT:                                                                      %
% xdot=[Vtdot,adot,bdot,thetadot,phidot,psidot,Qdot,Pdot,Rdot,Vn,Ve,Vz]%
%                                                                              %
% SYNOPSYS                                                                     %
% xdot=GSAM(x,u);                                                              %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [xdot]=GSAM(x,u);
global counter

% state
Vt=x(1); a=x(2); b=x(3); theta=x(4); phi=x(5); psi=x(6);
Q=x(7); P=x(8); R=x(9); Pn=x(10); Pe=x(11); Pz=x(12);
% controls
de=u(1); da=u(2); dr=u(3); thrust=u(4);

% Dimensions
s=0.365;      % wing area [m^2]
bb=1.46;      % wing span [m]
cc=0.25;      % wing mean chord [m]
CGh=1;        % CG heigth above ground [m]
CGd=0.25;     % CG position [m]
awi=0.035;    % wing incidence [rad]
m=2.3;        % mass [Kg]
Ixx=0.6;      % moment of inertia (X) [Kg*m^2]
Iyy=0.11;     % moment of inertia (Y) [Kg*m^2]
Izz=0.30;     % moment of inertia (Z) [Kg*m^2]
Ixz=0;        % moment of inertia (X/Z) [Kg*m^2]
```

```
% Lift
aML=0.297;    % alpha at maximum lift [rad]
CL0=0;        % lift at zero alpha
CLa=4.64;     % lift curve slope per radian [1/radian]
CLtail=0.40;  % coefficient of tail lift

% Drag
CD0=0.038;    % profile drag
CDCL=0.065;   % induced drag

% Sideforce
CYb=-0.52;    % sideforce from sideslip
CYdr=0.186;   % sideforce from rudder

% Pitching Moment
Cm0=0.072;    % pitching moment at zero alpha
Cma=-0.72;    % pitching moment with alpha
Cmde=-1.12;   % pitching moment with elevator
CmQ=-9.07;    % pitch rate damping
Cmadot=-3.63; % pitch damping with alpha rate

% Rolling Moment
Cldr=0.01;    % rolling moment with rudder
Clda=-0.35;   % rolling moment with aileron
Clp=-0.45;    % roll rate damping
Clb0=-0.02;   % rolling moment with beta
ClbCL=-0.05;  % rolling moment with beta variation due to CL
Clr0=0.0112;  % roll rate damping
ClrCL=0.16;   % roll rate damping variation due to CL

% Yawing Moment
Cnb=0.05;     % yawing moment with beta
Cnbdot=0.0;   % yawing moment with beta rate
Cndr=-0.04;   % yawing moment with rudder
Cnda=0;       % yawing moment with aileron
Cnp=0.013;    % yaw rate damping
Cnr0=-0.07;   % yaw rate damping
CnrCL=-0.0175;% yaw rate damping variation due to CL

% Gravity
g=9.81;       % gravity acceleration [m*s^2]

% Air density
rho=1.2;      % air density supposed constant

%Derived coefficients
rs=0.5*rho*s;
mg=m*g;
rsc=0.5*rho*s*cc;
rsb=0.5*rho*s*bb;
rsbb=0.25*rho*s*(bb^2);
rscc=0.25*rho*s*(cc^2);
I1=(Izz-Ixx);
I2=(Iyy-Izz-((Ixz^2)/Izz));
I3=((Ixz*Ixx-Iyy*Ixz)/Izz-Ixz);
I4=(Ixz/Izz);
I5=(Ixx-(Ixz^2)/Izz);
I6=(Ixx-Iyy-((Ixz^2)/Ixx));
```

```
I7=((Ixz*Iyy-Izz*Ixz)/Ixx-Ixz);
I8=(Ixz/Ixx);
I9=(Izz-(Ixz^2)/Ixx);


% computation of aerodinamic coefficients related to alpha

if(a>aML)
    CL=CL0+CLa*aML;
else
    CL=CL0+CLa*a;
end
CD=CD0+CDCL*(CL^2);
Clb=(Clb0+ClbCL*CL);
Clr=(Clr0+ClrCL*CL);
Cnr=(Cnr0+CnrCL*(CL^2));

% computation of the body axes speed
U=Vt*cos(a)*cos(b);
V=Vt*sin(b);
W=Vt*sin(a)*cos(b);

% computation of alpha_w as sum of alpha and wing incidence
aw=a+awi;

% aerodynamic forces drag lift sideforce computation (stability axes)
D=rs*(Vt^2)*CD;
Li=rs*(Vt^2)*(CL+CLtail*de);
S=rs*(Vt^2)*(dr*CYdr+b*CYb);

% body axes forces computation.
% Note the factors related to alpha used to convert the aerodynamic
% forces from stability to body axes.
% Note the factors related to psi,theta,phi used to convert
% the gravitational force from Euler body axes.
X=thrust-D*cos(a)+Li*sin(a)-mg*sin(theta);
Y=S+mg*sin(phi)*cos(theta);
Z=-Li*cos(a)-D*sin(a)+mg*cos(theta)*cos(phi);

% U',V',W' computation
Udot=(X/m)-(Q*W)+(R*V);
Vdot=(Y/m)-(R*U)+(P*W);
Wdot=(Z/m)-(P*V)+(Q*U);

% Vt',a',b' computation from U',V',W'
Vtdot=(U*Udot+V*Vdot+W*Wdot)/Vt;
adot=(U*Wdot-W*Udot)/(U^2+W^2);
bdot=(Vdot*Vt-V*Vtdot)/(Vt*(U^2+W^2)^0.5);

% psidot,thetadot,phidot (Euler angles) computation
phidot=P+Q*sin(phi)*tan(theta)+R*cos(phi)*tan(theta);
thetadot=Q*cos(phi)-R*sin(phi);
psidot=(Q*sin(phi)+R*cos(phi)/cos(theta));

% angular velocity computation (stability axes)
Pstab=P*cos(a)+R*sin(a);
Rstab=R*cos(a)-P*sin(a);
```

```
% aerodynamic moments computation (stability axes)
Mstab=rsc*(Vt^2)*(Cm0+Cma*aw+Cmde*de)+rscc*Vt*(CmQ*Q+Cmadot*adot);
Lstab=rsb*(Vt^2)*(Clb*b+Clda*da+Cldr*dr)+rsbb*Vt*(Clp*Pstab+Clr*Rstab);
Nstab=rsb*(Vt^2)*(Cnb*b+Cnda*da+Cndr*dr+Cnbdot*bdot)+...
          ...rsbb*Vt*(Cnp*Pstab+Cnr*Rstab);

% body axes moments computation.
% Note the factors related to alpha used to convert the aerodynamic
% moments from stability to body axes.
L=Lstab*cos(a)-Nstab*sin(a);
M=Mstab+L*(CGd-0.25)*cc*cos(a)+D*(CGd-0.25)*cc*sin(a);
N=Nstab*cos(a)+Lstab*sin(a);

% P',Q',R' (angular accelerations) computation
Qdot=(M+I1*R*P+Ixx*((R^2)-(P^2)))/Iyy;
Pdot=(L+I2*(Q*R)+I3*(P*Q)+N*I4)/I5;
Rdot=(N+I6*(P*Q)+I7*(Q*R)+L*I8)/I9;

% NAVIGATION
% Ve,Vn,Vz computation from U,V,W via DCM matrix
% definition of DCM coefficients
a11=cos(theta)*cos(psi);
a12=sin(phi)*sin(theta)*cos(psi)-cos(phi)*sin(psi);
a13=cos(phi)*sin(theta)*cos(psi)+sin(phi)*sin(psi);
a21=cos(theta)*sin(psi);
a22=sin(phi)*sin(theta)*sin(psi)+cos(phi)*cos(psi);
a23=cos(phi)*sin(theta)*sin(psi)-sin(phi)*cos(psi);
a31=-sin(theta);
a32=sin(phi)*cos(theta);
a33=cos(phi)*cos(theta);

%computation
Vn=U*a11+V*a12+W*a13;
Ve=U*a21+V*a22+W*a23;
Vz=U*a31+V*a32+W*a33;

% output vector
xdot=[Vtdot,adot,bdot,thetadot,phidot,psidot,Qdot,Pdot,Rdot,Vn,Ve,Vz];
\begin{small}
% increase the function call counter
counter=counter+1;
```

B.2 RK integration routine

```
function [newx,e]=RK23(x,in,dt);


% b21=1/5;
% b31=3/40; b32=9/40;
% b41=3/10; b42=-9/10; b43=6/5;
%
% c1s=-3/2; c2s=5/2;
%
% c1=19/54; c2=0; c3=-10/27; c4=55/54;
```

```
b21=1/4;
b31=15/169; b32=50/169;
b41=323375/314432; b42=-283075/78608; b43=1109485/314432;

c1s=-3/10; c2s=0; c3s=13/10;

c1=163/1950; c2=0; c3=9971/15150; c4=25432/98475;


k1=dt*WOT4(x,in);

k2=dt*WOT4(x+b21*k1,in);

k3=dt*WOT4(x+b31*k1+b32*k2,in);

k4=dt*WOT4(x+b41*k1+b42*k2+b43*k3,in);

newx=x+c1*k1+c2*k2+c3*k3+c4*k4;

newxs=x+c1s*k1+c2s*k2+c3s*k3;

e=max(abs((newx-newxs)./newx));

%e=(newx-newxs);
```

B.3 Trim routine

```
global x
%initial conditions
in=[0 0 0 4];
x=zeros(1,12);
x(2)=0.065;
x(4)=0.065;
%desired cruise speed
x(1)=13;
%trimming variables
s0=[x(1) x(2) in];
%simplex algorithm
[s,cost]=fminsearch('fcost',s0);
disp(['The obtained cost is ',cost]);
disp(['The suggested trim variables vare ',s]);


% cost function
function [cost]=fcost(s);
global x
x(2)=s(1);
x(3)=s(2);
x(4)=s(1);
in=s(3:end);
[xdot]=WOT4(x,in)
cost=[xdot(1:3) xdot(7:9)]*[xdot(1:3) xdot(7:9)]';
```

B.4 Linearization routine

```
function [A,B]=linearize(x,u);

tol=1e-6; %tolerance

%linearization of the A matrix

%set the initial dx step
dx=0.1*x;
 n=length(x);
 m=length(u);
for i=1:n
    if (dx(i)==0)
       dx(i)=0.1;
   end
end

last=zeros(n,1);
A=zeros(n,n);

for j=1:n
    xt=x;
    for i=1:11
        xt(j)=x(j)+dx(j);
        xd1=GSAM(xt,u); %derivative calculation at x+dx
        xt(j)=x(j)-dx(j);
        xd2=GSAM(xt,u); %derivative calculation at x-dx
        A(:,j)=(xd1-xd2)'/(2*dx(j));
        if max(abs(A(:,j)-last)./abs(A(:,j)+1e-12))<tol
            break      %check if the tolerance is satisfactory
        end
        dx(j)=0.5*dx(j); %reduce the step
        last=A(:,j);
    end
   iteration=i;
end


%linearization of the B matrix
last=zeros(n,1);
B=zeros(n,m);

%set the initial du step
du=0.1*u;
for i=1:m
    if (du(i)==0)
       du(i)=0.1;
   end
end

m=length(u);
for j=1:m
    ut=u;
    for i=1:15
        ut(j)=u(j)+du(j);
        xd1=GSAM(x,ut);  %derivative calculation at u+du
        ut(j)=u(j)-du(j);
```

```
        xd2=GSAM(x,ut);  %derivative calculation at u-du
        B(:,j)=(xd1-xd2)'/(2*du(j));
        if max( abs(B(:,j)-last)./abs(B(:,j)+1e-12))<tol;
            break  %check if the tolerance is satisfactory
        end
        du(j)=0.5*du(j); %reduce the step
        last=B(:,j);
    end
   iteration=i;
end
```

B.5 Actuator routine

```
function [udot]=AD(u,in);

global clamp

%some useful constants
RD=(180/pi);
DR=(pi/180);
sft=3.261188308459;

%define the actuator time constants
tau=[0.1 0.05 0.05 0.5];


%define the actuator limiters
uul=[25*DR,20*DR,30*DR,3*sft];
ull=[-25*DR,-20*DR,-30*DR,0];
udotl=[60*DR,80*DR,120*DR,3*sft/0.2];

udot=(in-u)./tau;

for cn=[1:1:length(u)],

    clamp(cn)=1;

    if (abs(udot(cn))>udotl(cn))
        udot(cn)=sign(udot(cn))*udotl(cn);
        clamp(cn)=0;
    end

    % control variable limitation
    if (u(cn)>=uul(cn))
        u(cn)=uul(cn);
        udot(cn)=min(0,udot(cn));
        clamp(cn)=0;
    elseif (u(cn)<=ull(cn))
        u(cn)=ull(cn);
        udot(cn)=max(0,udot(cn));
        clamp(cn)=0;
    end
end
```

B.6 Washout filter routine

```
function [wdot]=WI(w,input)
global tauw

%compute the derivative
wdot=[input(1:2),(input(3)-w(3))/tauw];
```

B.7 Dynamic feedback factors routine

```
function [kvi,kphi,kp,kr]=KKK(Vt)

%define the constant factors
ckvi = [ 0.000028864814473  -0.002145176319453   0.060964695614387...
 -0.466631817521348];

ckphi = [  0.00000367041654  -0.00037698916839   0.01457196716902...
 -0.25724700155499   1.89046121360699];

ckp = [0.00000203687711  -0.00021589537045   0.00860366459240...
  -0.15505371619899   1.10322512340311];

ckr = [0.00000817833847  -0.00080327077454   0.02925769888924...
  -0.47718264226502   3.20963323216599];

%calculate the speed power vector
vv=[Vt^4;Vt^3;Vt^2;Vt;1];

%compute the speed feedback
kvi= ckvi*vv(2:end,:);

%compute the phi feedback
kphi = ckphi*vv;

%compute the jaw feedback
kp = ckp*vv;

%compute the roll feedback
kr = ckr*vv;
```

Bibliography

[1] Xavier Riviere Bill Crowther. Flocking of autonomous unmanned air vehicles. 17^{th} UAV System conference, Bristol UK, April 2002.

[2] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioural model. *Computer Graphics*, 21(4):25–34, July 1987.

[3] Micropilot. Uavs autopilot producer. *http://www.micropilot.com.*

[4] Ian Kelly. Seven dwarfs. *http://www.coro.caltech.edu/People/ian/dwarfs.htm.*

[5] Adam Hayes. Moorebots. *http://www.coro.caltech.edu/Projects/Flocking/.*

[6] IEEE. Task force on cluster computing. *http://www.ieeetfcc.org.*

[7] Stelios Bounanos. A brief survey of systems for mobile computation. *http://gridswarms.essex.ac.uk/mobcompsys.html.*

[8] UCLA. The minuteman project. *http://www.icsl.ucla.edu/minuteman/.*

[9] MIT. The formation-flying autonomous blimps. *http://www.mit.edu/people/jhow/ff/blimps/blimps.html.*

[10] UWE Bristol. The flying flock. *http://www.ias.uwe.ac.uk/projects.htm.*

[11] Stanford University. The dragonfly project. *http://airtraffic1.stanford.edu/∼uav/.*

[12] *Standard Atmosphere.* U.S. Government Printing Office, Washington D.C., 1962.

[13] P.Henrici. *Discrete variable methods in ordinary differential equation.* J.Wiley & sons inc., 1962.

[14] W.H. Press S.A Teukolsky V.T. Vetterling B.P. Flennery. *Numerical Recipes in C: The art of scientific computing.* Cambridge University press, second edition, 1992.

[15] J.H. Verner. Families of embedded Runge-Kutta methods. *SIAM, Journal on numerical analysis*, 16(5):857–875, Oct 1979.

[16] B.L. Stevens F.L. Lewis. *Aircraft control and simulation.* J.Wiley & sons inc., second edition, 2003.

[17] Marc Rauw. Fdc 1.2 a simulink toolbox for fligth dynamics and control analysis. Technical report, May 2001.

[18] David F. Rogers. Turn performance sustained level turns.

[19] Shaw E. Schooling in fishes: Critique and review. *Natural History*, 84(8):4046, 1975.

[20] Xavier Riviere Bill Crowther. Rule-based guidance for flight vehicle flocking. Submitted to the Journal of Guidance, Dynamics and Control, September 2002.

[21] Arthur R. Marshall Jr. Garry D. Peterson. *1992 Lectures in complex systems*, chapter Animal aggregation: Experimental Simulation Using Vision-Based Behavioural Rules, pages 623–630. MA, Addison-Wesley, 1993.