"Multiobjective evolutionary-based optimization methods for trajectory planning of a quadrotor UAV"

ISBN 978-88-98382-01-9

Università degli Studi di Napoli
Federico II

Department of Industrial Engineering
School of Doctoral Research in Aerospace, Naval and Quality Engineering

Multiobjective evolutionary-based optimization methods for trajectory planning of a quadrotor UAV

Candidate
Egidio D'Amato

Tutor
Prof. Luigi de Luca

Supervisor
Prof. Giuseppe Del Core

Submitted on April 2013 for the XXV cycle

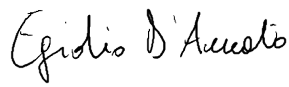
I, Egidio D'Amato, declare that this thesis titled, "Multiobjective evolutionary-based optimization methods for trajectory planning of a quadrotor UAV" and the work presented in it are my own work. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given and with the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

This PhD thesis has been defended in a public dissertation on May, 3rd 2013 under the judgement of a specialized commission composed by:

- Prof. Giorgio Guglieri, Department of Mechanical and Aerospace Engineering, Politecnico di Torino;
- Prof. Giulio Avanzini, Department of Innovation Engineering, Università del Salento;
- Prof. Salvatore Miranda, Department of Industrial Engineering, Università degli Studi di Napoli "Federico II".

April 2013
Department of Industrial Engineering
University of Naples Federico II
Naples



Abstract

This thesis describes the main research activity developed in a three years PhD program on flight dynamics. Optimization and UAVs flight control have been the main focus with methodological contributions on optimization, numerical and experimental work.

Unmanned Aerial Vehicles (UAV) captured the attention of both research and industrial worlds as a replacement for expensive human-piloted vehicles. In the last decade, they became widely used for several applications in which humans could be unnecessary or in some cases too in danger.

Many laboratories in the area of flight control, but also in the areas of robotics and control engineering in general, made significant research experiences on quadrotors.

A collaboration between University of Naples "Parthenope" and the Second University of Naples is aimed at designing and using UAVs for educational and research purposes. More than one quadrotor was built, tested in flight and used as a platform for testing flight control and navigation systems.

Several optimization problems may be encountered in the design of an UAV.

During the design phase, they arise from the choice of the hardware, the design and layout of the structure, the aerodynamics. On the other hand, for the Guidance Navigation and Control system, the management of single or fleets of UAVs requires the solution of many non-linear optimization problems. For this reason a multi-objective general purpose optimization software has been developed, integrating evolutionary methods, as genetic algorithm and ant colony, with game theory paradigms, as Nash and Stackelberg equilibria.

These methods have been primarily used to solve trajectory optimization problems with the scope of searching efficient flight trajectories in the presence of constraints.

The thesis is developed around the flight control of a quadrotor UAV. The following are the main steps of the work described in this thesis:

- dynamic and aerodynamic modelling oriented to flight control design;
- development of a distributed general purpose optimization software implementing Game Theory based paradigms and Ant Colony algorithm hybridization;
- Application of the above optimization methods to trajectory planning;
- Numerical simulations and flight experiments.

In Chapter 2, the quadrotor platform is described, together with the mathematical modelling and the design of the low level flight control system (attitude and speed control). In Chapter 3 the structure of the general purpose optimization software, mainly focused on the game theory layer and the ant colony algorithm is presented. In Chapter 4 the objectives of the optimization software are described and solved. Finally, in the Chapter 5, numerical simulations and flight tests are are shown.

Acknowledgments

I wish to thank the Universitá degli Studi di Napoli "Federico II" for the great possibility I received by this PhD and the Universitá degli Studi di Napoli "Parthenope" for funding my studies. In particular, I wish to thank the supervisor, Professor Giuseppe Del Core, for the useful advices and the experiences matured in classroom in front of students.

I would like to express my deep gratitude to Professor Massimiliano Mattei for the trust he had in me and for the responsibilities given to me. In particular, driving the flight dynamics laboratory has been a honour for me and I'm proud for this.

The Dipartimento di Ingegneria Industriale e dell'Informazione has been a home for me and I would like to express my very great appreciation to Professor Luigi Iuspa for the long talks and explanations on any argument, I learned and I enjoyed it very much. Thanks to Professor Luciano Blasi for the useful critiques always made with a smile and to all other professors that have never saved a greeting or a joke, making pleasant all working days.

I wish to thank Professor Lina Mallozzi, a guide for me in these three years. I learned a lot, she has been for me a "research mum" and I hope to be able to work together for a long time. Furthermore a thank to two colleagues, met during the PhD, Giovanni Petrone and Elia Daniele, for being part of the "game theory group".

I would also like to extend my thanks to the students, now graduated, that have been involved in the project; so thanks to Antonio Chiariello and Antimo Pontillo for the enthusiasm, the commitment and the friendship shown during their thesis and Salvatore Bassolillo and Giuseppe Scapaticci, because without them I would never finished in time.

Finally a special thank to a friend of mine Piero Prisco for the nice and original cover of this thesis.

Table of Contents

List of Figures

Chapter 1

Introduction

The Unmanned Aerial Systems (UAS) have always captured the attention of both research and industrial worlds as a replacement for expensive human-piloted vehicles. In the last decade, they have become widely used for several applications in which humans could be unnecessary or in some cases too in danger.

Thanks to the lack of dimensional specifications for human transport, these vehicles have extremely varying application domains in both indoor and outdoor environments. A specimen list may be:

- military patrol,
- civilian aerial surveillance,
- photogrammetry and professional photography and video making,
- low cost advertising,
- traffic monitoring,
- educational learning.

Obviously each application requires a specified UAS with different levels of control and manoeuvrability, may be needing high precision and/or high reliability.

1.1 History

The history of unmanned aircraft may be reduced to the history of all aircraft. In the past centuries there exist a lot of examples more or less creative of unmanned systems. One of the first users of a UAS could be the Chinese Genereal Zhuge Liang (180-234 AD) who used paper balloons fitted with oil-burning lamps to heat the air and to make enemies think there was a divine force at work. However, this is a very first and raw example not belonging to the modern era.

Historically unmanned aircraft followed an operational pattern described as the three D's: Dangerous, Dirty and Dull [1]. Dangerous where the life of the pilot may be at risk. Dirty where the environment may be contaminated by chemical, biological or radiological hazards. Dull where the task requires long hours, making the flight stressful and not desirable.

The first modern unmanned vehicle is the aerial torpedo. In 1916 the U.S. Navy funded Elmer Sperry to begin the development of an unmanned aerial torpedo. Sperry built a small, lightweight airplane that could be self-launched without a pilot, fly unmanned out to 1000 yards guided to a target and detonate its warhead at a point close enough to be effective against a warship. This idea, considering that the airplane had been just invented, was an incredible step ahead, because included battery-powered radio, electrical actuators and mechanical three-axis gyro-stabilization, all primitive technologies in that epoch. However Sperry's aerial torpedo was never put in service production, while another the following aerial bomb, thanks to Charles Kettering, was the first mass-produced unmanned aircraft. This demonstrated impressive distance and altitude performance and the validity of the airframe was proved with a manned model in a piloted flight.

After World War I, most of the work focused primarily on employing target drones as anti-aircraft weaponry. Unmanned aircraft technology played a key role in formulating air power doctrine and provided key data that contributed to America, England, and Japan concluding that aircraft carriers, which played such vital a role in upcoming World War II, were a good investment.

After the two World Wars, in Cold War years, unmanned aircraft development shifted toward reconnaissance and decoy missions. This trend has continued today where the most part of UAS are involved in data gathering. One of the first re-

connaissance high performance unmanned airplane was the Radio Plane YQ-1B, a high altitude target drone, modified to carry cameras. However, poor range and high cost were the reasons for cancellation of this program.

The U.S. Air Force pioneered the first mass-produced, long-range, high-speed unmanned aircraft designed to conduct primarily reconnaissance missions but evolved into a wide array of tasks from suppression of enemy air defenses to weapons delivery. The AQM-34 Lightning Bug has the longest service record for an unmanned aircraft. Designed as an initiative of the Ryan Aircraft Company in the late 1950s, the aircraft was powered by a turbojet, employed low drag wing and fuselage configuration and could reach altitudes in excess of 50,000 ft and speeds of 600 knots.

In the same years, the first VTOL UAS was developed: the Drone Anti-Submarine Helicopter (DASH) of U.S. Navy, born to extend the delivery range of antisubmarine homing torpedoes. The DASH used remote control via a pilot on the ship to take off and land, and then employed a gyro-stabilizer autopilot to reach a location.

From the very first unmanned aircraft, researchers have spent a lot of effort to gain independent flight operation. Requirements for maximum standoff distance, long endurance, and significant data streams from onboard sensors have followed technology improvements. With the advent of small, lightweight digital computers, inertia navigation technology, and finally the global positioning system (GPS) satellite network, UAS operation gained flight autonomy on par with a human-piloted vehicle.

Lightweight computer technology developed in the 1970s, which led to the worldwide explosion in computer science and digitalization, played the most significant role in UAS autonomy. With each advance in computing power unmanned aircraft gained great flexibility to weather conditions as well as new variables affecting the mission equipment payloads. Mapping data can be stored on-board, not only to improve navigation but also to enable a more accurate sensor camera imagery.

1.2 State of the Art

An Unmanned Aerial System (or Vehicle) is built on top of several elements that characterize its own features. The central part is the aircraft that permits to distinguish three first families of UAS based on fixed wing, rotor-wing (usually VTOL-Vertical TakeOff and Landing) or lighter-than-air vehicles.

Another kind of distinction may be done for the remotely piloted (called also Remotely Piloted Vehicle RPV) and auto-piloted UAS. The concept of autonomy is the ability for an unmanned system to execute its mission following a set of preprogrammed instructions without operator intervention from takeoff to touchdown. It's possible to find in literature the concept of aerobot to merge the concept of UAS and robot. This merging of two different worlds is due to an increasing research in autonomy for UAS and the great development of robotics in the last years. Furthermore, they share electronic hardware, software and control methods, thanks to the improves in reliability of robotics and more relaxed constraints typically requested for manned vehicles.

In the last decade many laboratories in the area of flight control, but also in the areas of robotics and control engineering in general, made significant research experiences on quadrotors, i.e. four rotor VTOL UAVs (Vertical Take-Off and Landing Unmanned Aerial Vehicles) also called quadcopters [2, 3, 4, 5, 6, 7]. Due to the cost reduction of components for propulsion and control, it is nowadays affordable to build a significant number of such flying platforms also for educational purposes.

The present thesis describes a quadrotor flight control design project born to build a new line of research of the Department of Industrial and Information Engineering (DIII) of the Second University of Naples (SUN) in collaboration with "Parthenope" University of Naples.

This experience clearly demonstrates that the development of a robotic platform [8] as the quadrotor can be a good starting point to improve laboratory facilities, to practically experience several aspects of flight control design and implementation: hardware and software design, model identification and validation, ground testing, control algorithms design, sensors integration, control algorithms implementation on embedded electronics, flight tests, and to get a proprietary platform for automatic control and flight dynamics researches.

In the last years many research group are working on this kind of vehicle as UAS testbeds for control algorithms [9, 10, 11, 12, 13].

Quadrotor aerial robotic vehicles have become a standard platform for robotics research worldwide. Low dimension, good maneuverability, simple mechanics and payload capability support a number of indoor and outdoor applications. As main drawback, the high power consumption can be mentioned. However, the trade-off is very positive and the improvements on battery technology will increase commercial opportunities.

Thanks to their features, as the high maneuverability, they enable safe and low-cost experimentation in mapping, navigation and control strategies for UAS moving in a three-dimensional space. From a robotic point of view, the ability to move in a 3D contex brings new research challenges compared to terrestrial robots.

Recently, there has been increasing interest in quadrotor. In literature there exists a lot of works on this kind of UAS, based on publicly available open-source projects or commercial ones. The relatively simple structure has attracted interest from accademia, industries and hobbyists. Unlike conventional helicopters, the lack of swashplates and the presence of four actuators, with reduced diameter of rotor, permits an easy building and doesn't need a constant maintenance.

Many research groups have designed their own quadrotors to suit particular specifications. X4-flyer, OS4, STARMAC, Pixhawk are some successful academic experiments. In the commercial world, Draganflyer, Gaui Quad flyer, X-UFO, Parrot AR-Drone and ASCTEC Firefly are very famous and the most used. Furthermore, open source quadrotors are emerging with contributions from academia, hobbyists and corporations. The open source code shared on the web allows very fast development process, because debug and new features can be tested by other people in the community. Some important examples are Arducopter, Openpilot, Paparazzi and Mikrokopter.

From a control point of view, several control techniques are used for quadrotor UAS. In particular in [14, 15, 16, 17] thanks to Lyapunov Theory, it is possible to ensure, under certain condition, the asymptotical stability of the helicopter. The classic PID structures [17, 18, 19] doesn't require specific model parameters and the

control law is simpler to implement. To improve performance under uncertainties and unmodelled dynamics adaptive techniques can be implemented as in [20, 21]. Linear Quadratic Regulator (LQR) in [16, 17] shows that the main advantage of this technique is that the optimal input signal turns out tobe obtainable from full state feedback (by solving the Riccati equation). On the other hand the analytical solution to the Riccati equation is difficult to compute. Backstepping control technique [22, 22, 23] guarantee the convergence of the quadrotor internal states but at a great computational cost. Furthermore other new control algorithms done with fuzzy techniques [24, 25], neural networks [26] and reinforcement learning [27] can be used.

Another scope of quadrotor design may be the building of multi-agent systems, to increase robustness and flexibility, typical of multiple-robot systems. The use of multiple unmanned aerial vehicles combines these benefits with agility and pervasiveness of aerial platforms.

The degree of autonomy of a multi-UAV system should be tuned according to the specificities of environment and objective of mission. Fully autonomous UAV systems are not often appropriate, because, in general, the use of semi autonomous groups of UAVs, supervised or partially controlled by human operators, is the only feasible solution to deal with the complexity and unpredictability of real-world scenarios, for instance search and rescue missions, exploration of large environments and those high-risk situations in which only a human can take the responsibility of a critical decision.

Chapter 2

Quadrotor Dynamic Modelling and Flight Control Design

2.1 Introduction

In this chapter, the non-linear dynamic system equations are derived. The used modelling approach consists in writing mathematical equations that describe the movements and dynamics of a quadrotor UAS as derived from the governing laws and principals of classical physics. The set of non-linear dynamic system equations that describe the physical behaviour of the quadrotor are implemented in a simulated environment to design the flight control system and test it before the implementation on the embedded hardware.

2.2 Basic concepts

Quadrotor is propelled by four rotors in a cross configuration. The cross structure is quite thin and light, but it shows robustness by linking mechanically the motors. In the configuration used in this thesis, each propeller is connected directly to the motor shaft, without using reduction gears. All the propellers axes of rotation are fixed and parallel, with fixed-pitch blades.

Consider a rigid model composed of a thin cross structure with four propellers on its tips. The front and the rear propellers rotate counter-clockwise, while the

left and the right ones turn clockwise. This configuration of opposite pairs directions removes the need for a tail rotor (needed instead in the standard helicopter structure). Figure 2.1 shows the structure model in hovering condition, where all the propellers have the same speed.

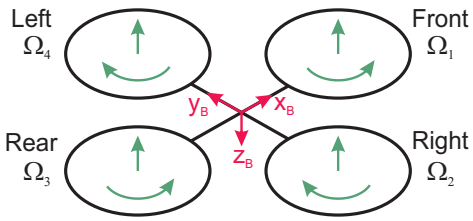
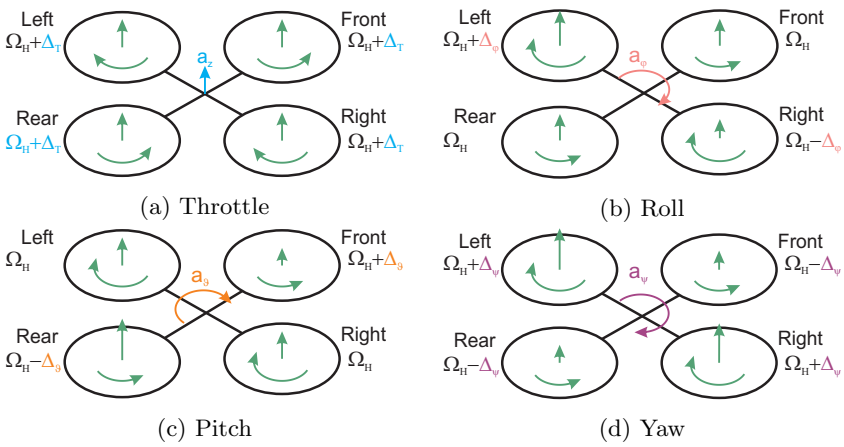


Figure 2.1: Hovering quadrotor scheme

The fixed-body reference system is represented in red colour. For each rotor, two arrows are drawn: one for the thrust and the curved one for the direction of rotation. In the model of figure 2.1 all the propellers rotate at the same speed, counterbalancing the weight of the UAV. Thus, the quadrotor performs a steady-state flight called hovering.

The quadrotor is an under-actuated force-controlled vehicle with six DoF, involving strong nonlinear dynamics and coupling. However, thanks to its structure, it is quite easy to choose the four best controllable variables and to decouple them to make the controller easier. The four basic movements, targets of the control system, are related to its altitude and attitude as shown in figure 2.2:

- throttle command is provided by increasing (or decreasing) all the propeller speeds by the same amount. It leads to a vertical force in the body-fixed frame;
- roll command is provided by an increase (decrease) of the left propeller speed and a decrease (increase) of right one. It leads to a torque around the x_B axis which makes the queadrotor turn;
- pitch command is similar to the roll one but acts on front and rear rotors. It leads to a torque around the y_B axis;
- yaw command is provided by increasing (or decreasing) the front-rear couple of propellers and by decreasing (or increasing) simultaneously the lef-right one. It provides a torque with respect to the z_B axis.



(a) Throttle (b) Roll

(c) Pitch (d) Yaw

Figure 2.2: Quadrotor basic movements

2.3 Quadrotor hardware design

In this section, the on-board system architecture and the structural layout of the quadrotor are shown.

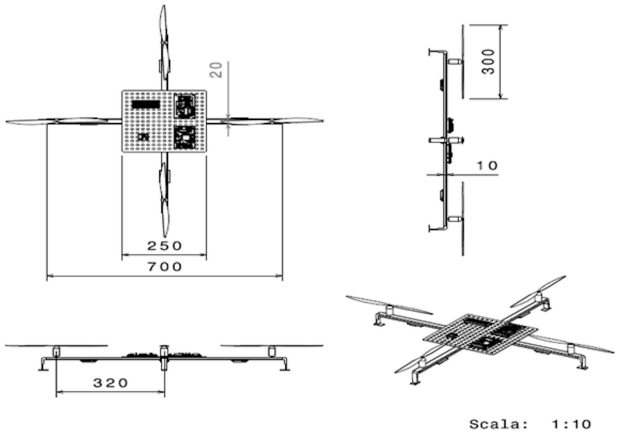
2.3.1 Quadrotor structure

Two structural layout have been designed, built and developed, for flight tests.

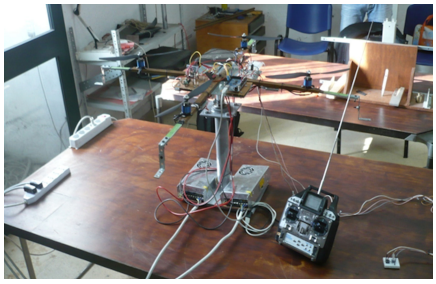
The first design is the simplest, only made by two orthogonal aluminium arms using rectangular cross-section beams and a central aluminium plate for placing electronics. Based on this design two prototypes were built; the first is constrained to ground, having only three DoF: roll, pitch and yaw; the motion is guaranteed by an universal joint plus a bearing to join the vehicle to a fixed ground base. It was a test-bench for hardware and software implementation and for a first identification of the model.

In Table 2.1 physical characteristics of first quadrotor designs (constrained and not) are shown.

The second structure was designed to strengthen several wake points of the first design: weight, little and unsecured place for electronics, landing gear. In figure 2.4 the cad drawing and the photo of the first prototype are shown and in table 2.2 physical characteristics of the second design are summarized.



(a) CAD Drawings



(b) Constrained test-bench

(c) Free flying

Figure 2.3: First quadrotor layout

2.3.2 On-board and ground systems

The electronic hardware was split in two parts: an on-board embedded electronics and a ground guidance system for telemetry and safety controls.

The on-board electronics is one of the most important components of the quadcopter and there is no way that the quadcopter can even be flown steadily without it. It is responsible for attitude control system and for autopilot tasks.

It is itself divided into a low-level integrated electronics, with sensors for attitude control, that provides stability to the vehicle, and a high level CPU implementing the autopilot and communication functions with the ground station. The main guidance functions are implemented on the ground station. In Figure 2.5 a scheme

l	$0.33m$	distance between thrust axis and center of gravity
m	$2.50kg$	Quadrotor mass
I_{xx}	$0.077kgm^2$	Inertial moment with respect to the center of gravity
I_{yy}	$0.077kgm^2$	Inertial moment with respect to the center of gravity
I_{zz}	$1.012kgm^2$	Inertial moment with respect to the center of gravity
I_{xy}	$\sim 0kgm^2$	Inertial product with respect to the center of gravity
I_{xz}	$\sim 0kgm^2$	Inertial product with respect to the center of gravity
I_{yz}	$\sim 0kgm^2$	Inertial product with respect to the center of gravity
d	$0.045m$	Distance between center of gravity and center of rotation for constrained quadrotor
I'_{xx}	$0.077kgm^2$	Inertial moment with respect to the center of gravity for constrained quadrotor
I'_{yy}	$0.077kgm^2$	Inertial moment with respect to the center of gravity for constrained quadrotor
I'_{zz}	$1.012kgm^2$	Inertial moment with respect to the center of gravity for constrained quadrotor

Table 2.1: Quadrotor characteristics

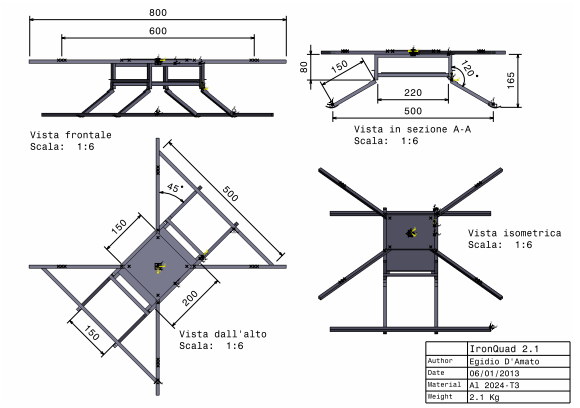
l	$0.30m$	distance between thrust axis and center of gravity
m	$2.10kg$	Quadrotor mass
I_{xx}	$0.020kgm^2$	Inertial moment with respect to the center of gravity
I_{yy}	$0.020kgm^2$	Inertial moment with respect to the center of gravity
I_{zz}	$0.330kgm^2$	Inertial moment with respect to the center of gravity
I_{xy}	$\sim 0kgm^2$	Inertial product with respect to the center of gravity
I_{xz}	$\sim 0kgm^2$	Inertial product with respect to the center of gravity
I_{yz}	$\sim 0kgm^2$	Inertial product with respect to the center of gravity
d	$-0.040m$	Distance between center of gravity and center of rotation

Table 2.2: Second quadrotor design characteristics

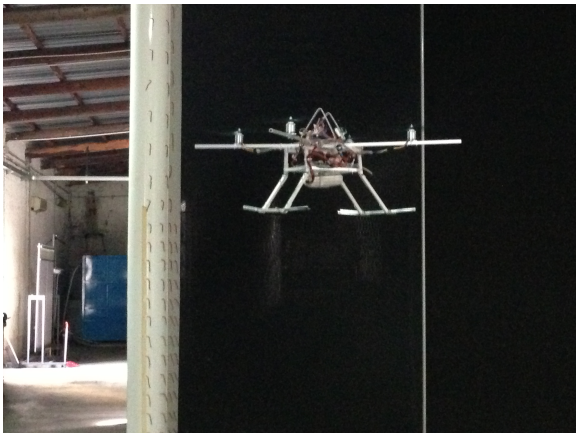
of the electronic architecture is presented.

Due to low-cost objectives, the on-board hardware target should be cheap and largely available on the market. In this thesis, a dsPIC33 was used as CPU for the low-level electronics. Thanks to its DSP (Digital Signal Processor) integrated capabilities, sensor integration and filtering can implemented in a very efficiently way.

To speed-up the first development, filtering and sensor fusion are based on DCM algorithm. All the mathematical principles involved have been studied and developed in Mahonyś papers [28] and [29]. The draft paper by William Premerlani DCM Imu Theory [30] has been used as a basis for the Sensor Fusion Module DCM implementation.



(a) CAD drawings



(b) Prototype photo

Figure 2.4: Second quadrotor design

Cheap MEMS inertial sensors turn out to be appropriate to the attitude control purposes [31]. A three axis accelerometer, MMA7361L, a two axes gyroscope (for roll and pitch), IDG500, and a single axis gyroscope (for yaw), ISZ500, plus a three axes magnetometer, HMC5843 were used. An EM-406 GPS was also used to implement autopilot functions on-board.

The high level part of embedded electronics, used for navigation purposes, was assigned to an ARM9 CPU board. In particular the Pandaboard open source hardware was used which is equipped with an OMAP4430 dual core ARM Cortex A9 CPU, 1GB of system RAM, integrated Wi-Fi and bluetooth.

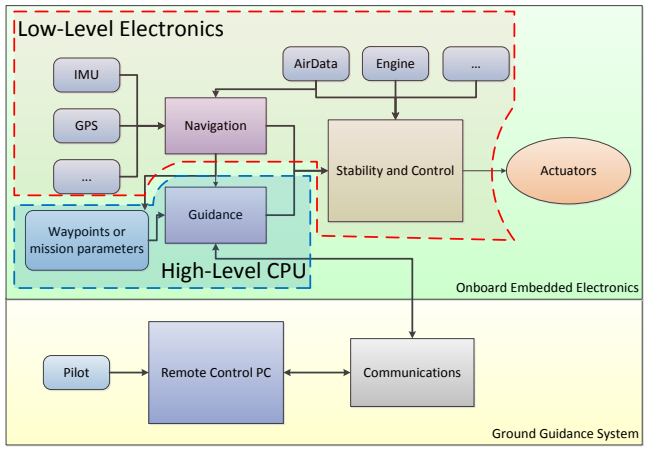


Figure 2.5: Electronic architecture design

The use of an higher level control layer, with this plenty of peripherals, and the use of a complete operating system, based on a linux[1] distribution, also allowed to easily implement communication functions with the ground station using WI-FI integrated hardware and already available operating system protocols. Furthermore it makes the UAV easily ready for further expansions.

2.3.3 Actuation System

Vehicle actuation system is derived by aero-modeling typical hardware. 20A-11.1V brushless motors with 12 inches rotors were used. Tests on the motor-rotor couple have guaranteed a fixed-point maximum thrust for a single axis of about 2 kg. This kind of motors needs a regulator to operate. Four 30A regulators connected to a 3-cell 5000 mAh Lipo battery were mounted on board to feed motors. With such a quadrotor configuration a flight of about 10 minutes with a take off weight of about 2 kg is guaranteed. For on-ground test bench, two power supplies were used to feed motors to avoid stressing batteries during tests.

A static motor+rotor model identification was performed to find the link between input and output of the actuation system. Neglecting rotor dynamics, the expression of the force and axial momentum generated by each motor depends on

[1] Ubuntu server 12.04 was used

the square of the propeller angular speed via aerodynamic coefficients, but to linearize the system and make it directly linked to the hardware platform, it's possible to relate thrust and momentum of each motor with the input signal of brushless motor regulator as following:

$$F_i = k_T PWM_i + c_0 \tag{2.1}$$

$$M_i = k_R PWM_i + c_1 \tag{2.2}$$

The coefficients k_T and k_R may be extracted by performing a load test using a calibrated balance. In this thesis a three axes strain gauge balance was used. In Figures 2.6, 2.7 and 2.8 the scheme, the test-bench and the results of the motor tests are shown.

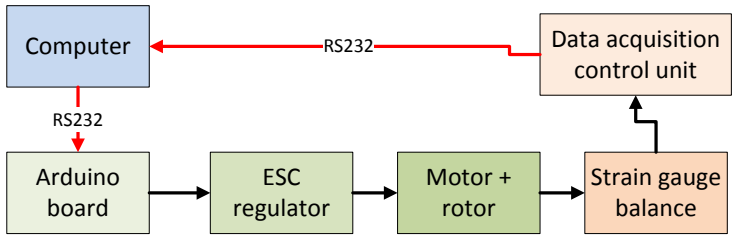


Figure 2.6: Motor tests scheme

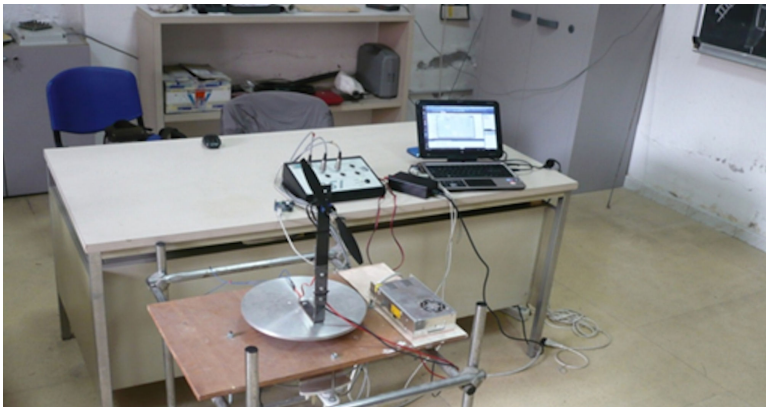


Figure 2.7: Motor tests results

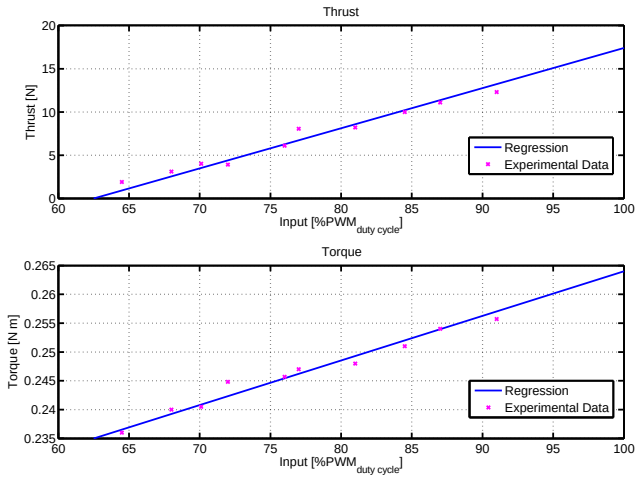


Figure 2.8: Thrust-PWM and Torque-PWM relationships

2.4 System Modelling

In this paragraph physical the six DoF motion equations are described, where the main physical quadrotor properties are extracted from the CAD model. An open-cycle identification of an unstable system is very difficult, so the system was divided in several sub-models to design a preliminary stabilization system.

The model developed is based on a rigid and symmetrical structure, with a standard aerodynamic model.

2.4.1 Reference Systems

If we think at the six DoF motion, the quadrotor can be modelled as a rigid body built on two orthogonal arms, with four independent rotors at their tips. Each couple of rotors rotates in the opposite direction to each other, in order to reduce the gyroscopic actions and the aerodynamic counter-torque around the z-body axis.

In order to establish the dynamic model of the quadrotor two reference frames have to be defined, which are the earth inertial frame $(OXYZ)_1$ and the body-fixed frame $(Oxyz)_B$ [32] respectively. Due to the low speed of the flight, the absence of wings, and the small dimension of rotors, the use of a wind-axes frame can be avoided.

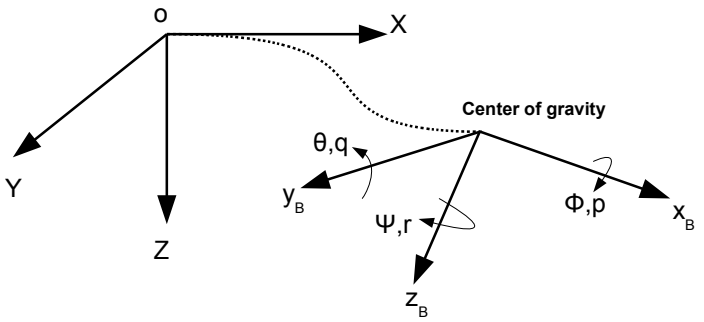


Figure 2.9: Reference systems

The origin of quadrotor coordinate system is its center of gravity, with x_B coinciding with one of the arms, y_B with the other and z_B perpendicular to the plane Ox_By_B and directed downwards.

The inertial frame, with origin coincident to the center of gravity of the vehicle at start, Z_1 axis opposite to the acceleration of gravity, X_1 in the direction of the initial heading, and Y_1 perpendicular to the plane OX_1Z_1, can be used.

The attitude of the vehicle refers to the inertial frame and it is defined by a set of three rotations, called Euler's angles (ψ, θ, ψ), to make the body frame parallel to the inertial reference system.

The rotation matrix is:

$$R_{321} = R_\psi R_\theta R_\phi = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi - c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \tag{2.3}$$

where $c(\cdot)$ and $s(\cdot)$ are respectively the trigonometric functions cos and sin.

2.4.2 Rigid body equations

Consider F_X, F_Y, F_Z components of force vector $\underline{F}$, M_X, M_Y, M_Z components of momentum vector $\underline{M}$, U, V, W components of speed vector $\underline{v}$, in the inertial frame; p, q, r are components of the angular velocity vector Ω of the body reference

system with respect to the inertial frame. Newton's second law can be written as:

$$\underline{F} = \frac{d(m\underline{v})}{dt} = m\frac{d\underline{v}}{dt} \tag{2.4}$$

$$\underline{M} = \frac{d\underline{h}}{dt} = \frac{d(I\underline{\Omega})}{dt} \tag{2.5}$$

where d/dt is the time derivative with respect to the inertial reference frame, tha angular momentum is $\underline{h} = I\underline{\Omega}$, the inertial matrix is:

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \tag{2.6}$$

Euler's equation of motion expressed in the body reference system are given by 2.7.

$$\begin{aligned} F_x &= m(\dot{U} + qW - rV) \\ F_y &= m(\dot{V} + rU - pW) \\ F_z &= m(\dot{W} + pV - qU) \\ L &= I_{xx}\dot{p} - I_{yz}(q^2 - r^2) - I_{xz}(\dot{r} + pq) - I_{xy}(\dot{q} - rp) - (I_{yy} - I_{zz})qr \\ M &= I_{yy}\dot{q} - I_{zx}(r^2 - p^2) - I_{xz}(\dot{p} + qr) - I_{yz}(\dot{r} - pq) - (I_{zz} - I_{xx})rp \\ N &= I_{zz}\dot{r} - I_{xy}(p^2 - q^2) - I_{yz}(\dot{q} + rp) - I_{zx}(\dot{p} - qr) - (I_{xx} - I_{yy})pq \end{aligned} \tag{2.7}$$

Finally the kinematic equations to derive the trajectory of the quadrotor in the inertial frame can be determined starting from u, v, w components of speed vector in the body reference system (2.8).

$$\begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = R_\psi R_\theta R_\phi \begin{Bmatrix} U \\ V \\ W \end{Bmatrix} = R_{321} \begin{Bmatrix} U \\ V \\ W \end{Bmatrix} \tag{2.8}$$

In particular:

$$\begin{aligned} u = \tfrac{dx}{dt} &= Uc\psi c\theta + V(c\psi s\theta s\phi - s\psi c\theta) + W(c\psi s\theta c\phi + s\psi s\phi) \\ v = \tfrac{dy}{dt} &= Us\psi c\theta + V(s\psi s\theta s\phi - c\psi c\theta) + W(s\psi s\theta c\phi - c\psi s\phi) \\ w = \tfrac{dz}{dt} &= -Us\theta + V(c\theta s\phi) + W(c\theta c\phi) \end{aligned} \tag{2.9}$$

According to the kinematic relationship between Euler angles and angular velocity, the equations 2.10 complete the set of twelve needed to describe the rigid body motion in space.

$$\begin{aligned}\dot{\phi} &= P + Q\sin\phi\tan\theta + R\cos\phi\tan\theta \\ \dot{\theta} &= Q\cos\phi + R\sin\phi \\ \dot{\psi} &= (Q\sin\phi + R\cos\phi)\sec\theta\end{aligned} \tag{2.10}$$

External forces and moments are generated by gravity, aerodynamics and propulsion. In first approximation, the expression of the force and axial momentum generated by each motor depends on the square of the propeller angular speed via aerodynamic coefficients. Considering k_T and k_R respectively force and axial momentum coefficients and l the distance between rotors axes, applied forces and moments on quadrotor can be modelled as in 2.11.

$$\begin{aligned}F_z = \textstyle\sum_{i=1}^{4} F_i = \sum_{i=1}^{4} k_T\omega_i^2 = 4k_T\omega_i^2 \\ M_x = (\omega_2^2 - \omega_4^2)k_T l \\ M_y = (\omega_3^2 - \omega_1^2)k_T l \\ M_z = \textstyle\sum_{i=1}^{4} M_i(-1)^{i+1} = \sum_{i=1}^{4} \omega_i^2 k_R(-1)^{i+1}\end{aligned} \tag{2.11}$$

In matrix notation:

$$\begin{Bmatrix} F \\ M_x \\ M_y \\ M_z \end{Bmatrix} = \begin{bmatrix} k_T & k_T & k_T & k_T \\ 0 & k_T l & 0 & -k_T l \\ -k_T l & 0 & k_T l & 0 \\ k_R & -k_R & k_R & -k_R \end{bmatrix} \begin{Bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{Bmatrix} \tag{2.12}$$

Gravitational force acts on the center of gravity and it's defined in the body-fixed frame by:

$$\begin{aligned}\underline{F_g} = \begin{Bmatrix} F_{gx} \\ F_{gy} \\ F_{gz} \end{Bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{Bmatrix} 0 \\ 0 \\ mg \end{Bmatrix} \\ &= \begin{Bmatrix} -mg\sin\theta \\ mg\cos\theta\sin\phi \\ mg\cos\theta\cos\phi \end{Bmatrix}\end{aligned} \tag{2.13}$$

As for the aerodynamic forces and moments, drag is opposite to the speed vector and depends on the dynamic pressure and the shape of the structure. The

aerodynamic drag can be considered focused on the center of gravity of the quadrotor and it depends on C_d and S, related to the quadrotor geometry. However in a first approximation it can be neglected if low speed manoeuvres are requested to the vehicle. This is a conservative assumption due to the damping nature of aerodynamic forces and moments: [2]

$$\underline{D} = -0.5\rho \left|V\right|^2 SC_d\underline{v} \tag{2.14}$$

2.4.3 Simulator Implementation

The mathematical model of the aerial vehicle is very useful to study the dynamical response and to design the control system. Euler's equations, summarized in 2.15, can be implemented in a simulation environment and numerically solved. For this purpose, it's been used the software Simulink© part of Matlab© suite.

$$\begin{gathered}
F_x = m(\dot{U} + qW - rV) \\
F_y = m(\dot{V} + rU - pW) \\
F_z = m(\dot{W} + pV - qU) \\
L = I_{xx}\dot{p} - I_{yz}(q^2 - r^2) - I_{xz}(\dot{r} + pq) - I_{xy}(\dot{q} - rp) - (I_{yy} - I_{zz})qr \\
M = I_{yy}\dot{q} - I_{zx}(r^2 - p^2) - I_{xz}(\dot{p} + qr) - I_{yz}(\dot{r} - pq) - (I_{zz} - I_{xx})rp \\
N = I_{zz}\dot{r} - I_{xy}(p^2 - q^2) - I_{yz}(\dot{q} + rp) - I_{zx}(\dot{p} - qr) - (I_{xx} - I_{yy})pq \\
\dot{\phi} = P + Q\sin\phi\tan\theta + R\cos\phi\tan\theta \\
\dot{\theta} = Q\cos\phi + R\sin\phi \\
\dot{\psi} = (Q\sin\phi + R\cos\phi)\sec\theta \\
u = \tfrac{dx}{dt} = Uc\psi c\theta + V(c\psi s\theta s\phi - s\psi c\theta) + W(c\psi s\theta c\phi + s\psi s\phi) \\
v = \tfrac{dy}{dt} = Us\psi c\theta + V(s\psi s\theta s\phi - c\psi c\theta) + W(s\psi s\theta c\phi - c\psi s\phi) \\
w = \tfrac{dz}{dt} = -Us\theta + V(c\theta s\phi) + W(c\theta c\phi)
\end{gathered} \tag{2.15}$$

The mathematical model in the state-space form can be formulated in the following way:

$$\begin{aligned}
\dot{x} &= f(x,u) \\
y &= g(x,u)
\end{aligned} \tag{2.16}$$

Assuming the following vectors of states x, inputs u and outputs y:

[2] In this thesis, aerodynamic drag is considered only in the full simulator implementation

$$\underline{x} = \begin{bmatrix} P & Q & R & \phi & \theta & \psi & U & V & W & X & Y & Z \end{bmatrix}^T \tag{2.17}$$

$$\underline{u} = \begin{bmatrix} U_z & U_\phi & U_\theta & U\psi \end{bmatrix}^T \tag{2.18}$$

$$\underline{y} = [x]^T \tag{2.19}$$

In figure 2.10 is presented the Simulink quadrotor control block.

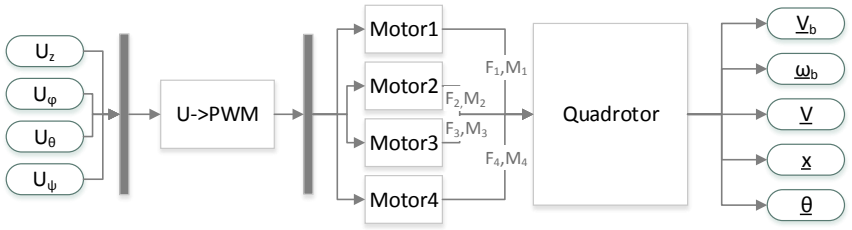


Figure 2.10: Quadrotor Mask

This is built on the quadrotor physics block that presents rotor forces and torques as inputs.

Motor blocks include actuator dynamics, so they have PWM signal as input and rotor force and torque as output. The link between themselves is due to k_T and k_R coefficient (as in par.2.3.3), using a first order transfer function with time constant τ_A:

$$F = \frac{k_T l}{\tau_A s + 1} \tag{2.20}$$

Furthermore, ESC regulators, needed to drive brushless motors, show a delay to read PWM input signal.

Forces and moments on the quadrotor can be evaluated through the following relationships:

$$\begin{Bmatrix} F_z \\ M_x \\ M_y \\ M_z \end{Bmatrix} = FG \begin{Bmatrix} PWM_1 \\ PWM_2 \\ PWM_3 \\ PWM_4 \end{Bmatrix} = F \begin{Bmatrix} U_z \\ U\phi \\ U\theta \\ U\psi \end{Bmatrix} \tag{2.21}$$

Where:

$$F = \begin{bmatrix} k_T & 0 & 0 & 0 \\ 0 & k_T l & 0 & 0 \\ 0 & 0 & k_T l & 0 \\ 0 & 0 & 0 & k_R \end{bmatrix} \tag{2.22}$$

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 1 & -1 & 1 & -1 \end{bmatrix} \tag{2.23}$$

However, to use a physical block with explicit forces and torques for each rotor, the following relationship is needed:

$$\begin{Bmatrix} F_z \\ M_x \\ M_y \\ M_z \end{Bmatrix} = F^* \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ M_1 \\ M_2 \\ M_3 \\ M_4 \end{Bmatrix} = F^* G^* \begin{Bmatrix} PWM_1 \\ PWM_2 \\ PWM_3 \\ PWM_4 \end{Bmatrix} = F^* G^* F^- 1 \begin{Bmatrix} U_z \\ U\phi \\ U\theta \\ U\psi \end{Bmatrix} \tag{2.24}$$

Where:

$$F^* = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & l & 0 & -l & 0 & 0 & 0 & 0 \\ -l & 0 & l & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \end{bmatrix} \tag{2.25}$$

$$G^* = \begin{bmatrix} k_T & 0 & 0 & 0 \\ 0 & k_T & 0 & 0 \\ 0 & 0 & k_T & 0 \\ 0 & 0 & 0 & k_T \\ k_R & 0 & 0 & 0 \\ 0 & k_R & 0 & 0 \\ 0 & 0 & k_R & 0 \\ 0 & 0 & 0 & k_R \end{bmatrix} \tag{2.26}$$

2.4.4 Model Identification

The model identification activities can be divided in three groups:

- measurements of the measurable parameters, weight, inertia matrix, aerodynamic coefficient in a wind tunnel, thrust coefficients;
- on ground dynamic model identification;
- In flight dynamic model identification.

Dynamic models can be of different types depending on their use. Among parameters measurement activities, weight is easy to extract by weighing the hardware. Inertia matrix may be computed using a geometric model of quadrotor with the weight of each component and its distance from the center of gravity. The throttle and torque coefficients are extracted using a strain gauge balance (see par.2.3.3), considering the relationship between PWM and applied forces as shown in eq.2.1, 2.2 and Figure 2.8.

The open loop instability of the vehicle requires to design a preliminary stabilization system prior to flight tests for the complete model calibration. Assuming that this stabilization system can be designed with SISO actions on the three axes the first step is to identify the two rotational dynamics around x_B (y_B being equivalent) and z_B body axes. A ground test facility was used for this purpose.

The on ground experimental platform shown in figure 2.3b has got three rotational DoF, namely roll, pitch and yaw [33]. In this on ground constrained configuration, the quadrotor rotates around a point which is on the Z axis below the center of gravity. For this reason it behaves as an inverse pendulum. Equations for pitch, roll and yaw motion become:

$$\begin{aligned} \ddot{\phi} &= \tfrac{k_T l}{I_{xx}} U_\phi - \tfrac{cd_p}{I_{xx}} \dot{\phi} + \tfrac{mgd}{I_{xx}} \sin\phi \\ &= k_{1x} U_\phi - k_{2x} \dot{\phi} + k_{3x} \sin\phi \end{aligned} \tag{2.27}$$

$$\begin{aligned} \ddot{\theta} &= \tfrac{k_T l}{I_{yy}} U_\theta - \tfrac{cd_q}{I_{yy}} \dot{\theta} + \tfrac{mgd}{I_{xx}} \sin\theta \\ &= k_{1y} U_\theta - k_{2y} \dot{\theta} + k_{3y} \sin\theta \end{aligned} \tag{2.28}$$

$$\begin{aligned} \ddot{\psi} &= \tfrac{k_R}{I_{zz}} U_\psi - \tfrac{cd_r}{I_{zz}} \dot{\psi} \\ &= k_{1z} U_\psi - k_{2y} \dot{\psi} \end{aligned} \tag{2.29}$$

The three equations turn out to be decoupled and, if rotation on all the axes except one are locked, experiments can be made to identify the model parameters

on the axis free to rotate.

Open loop experiments consist in assuming a certain initial attitude angle, and giving a momentum step to force a rotation. Having a certain number of experimental traces and parameters $k_{i[x,y,z]}$ to identify, these are found, as usual in parametric identification [34] minimizing the mean squared difference between the model response and experimental values of the output.

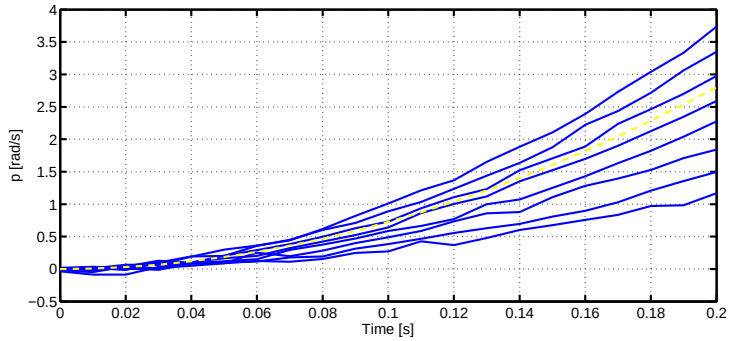


Figure 2.11: Open-loop identification on angular velocity

Figure 2.11 shows a family of experimental responses on the roll angular speed that are used to tune the model giving the response marked in yellow. However the unstable nature of the quadrotor makes very difficult to have a precise evaluation of the model parameters. Furthermore, the use of joints and bearings causes frictional forces that are dominant over the low speed aerodynamics.

2.5 Control system design

2.5.1 Introduction

The main objectives of a basic quadrotor flight control system are the stabilization of the platform, the attitude control and the autopilot. A good control system must ensure in all operating conditions:

- closed-loop stability,
- regime good performance,
- dynamical good performance,

- robustness to system and environment uncertainty,
- bounded control signals.

The model developed in Section 2.4 describes the differential equations of the system. The aim is to supply the motors in such a way to manage the quad in the desired position. This process is also known as inverse kinematics and inverse dynamics. The quadrotor dynamics must be simplified to provide an easy inverse model which can be implemented in the control algorithms.

In particular:

- most of the angular contributes come from cross coupling of angular speeds (gyroscopic effects). Since the motion of the quadrotor can be assumed close to hovering, small angular changes occur and so they can be neglected;
- angular accelerations are referred to the angles of the quadrotor in the body-fixed reference system. In the hovering condition, the transfer matrix to determine the attitude in the inertial frame is close to the identity matrix and so the acceleration equations can be referred directly to the Euler angle accelerations;
- aerodynamic drag is a damping term and in a first control design may be neglected.

In this thesis a model-based design of the control system was built, starting from the decoupled mathematical model. The first step was to design a PID controller to stabilize the on-ground constrained platform and to perform a first closed-loop fine tuning of the model. After this step, the developed 3 DoF PID controller was ported on the flying structure and expanded with the autopilot part of the control system.

2.5.2 PID controller

PID is surely one of the most used linear regulator in the industrial area. Some reasons of this success are:

- dimple structure,
- good performance,
- easy tuning also without a specific model of the target system.

There are several approaches to control such a quadrotor, from the classical system based on a standard PID controller to very complex techniques. In this work a feedback PID control approach was used, because it's strictly physics-related and easy to fine tune.

The standard PID algorithm may be described by the following equation:

$$u(t) = k_P(e(t) + \frac{1}{T_i}\int_0^t e(\tau) + d\tau + T_d\frac{de(t)}{dt}) \tag{2.30}$$

Where y is the measured process variable, r the reference, u is the control signal and e is the measured error. As summarized in figure 2.13 the control signal is the sum of three terms: the P-term (proportional to the error), the I-term (proportional to the integral of the error) and the D-term (proportional to the integral of the error).

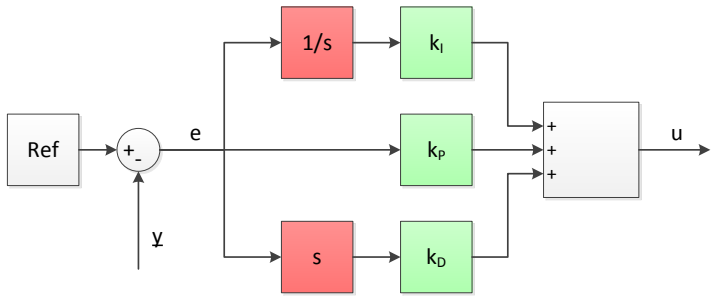


Figure 2.12: Traditional PID structure

The transfer function from measurement y to controller output u of a PID controller is:

$$C(s) = -k_P(1 + \frac{1}{sT_I} + sT_D) = -k_P(1 + \frac{k_I}{s} + sk_D) \tag{2.31}$$

The controller parameters, to tune during the controller design, are the proportional, integral and derivative gains, k_P, k_I and k_D.

In figure 2.14 the implemented quadrotor PID algorithm is shown. According to hardware definition (see 2.3.2), it's divided in two layers: the low-level control system to perform attitude stabilization and the high level control system to act as autopilot. It's important to underline that the low level control system may

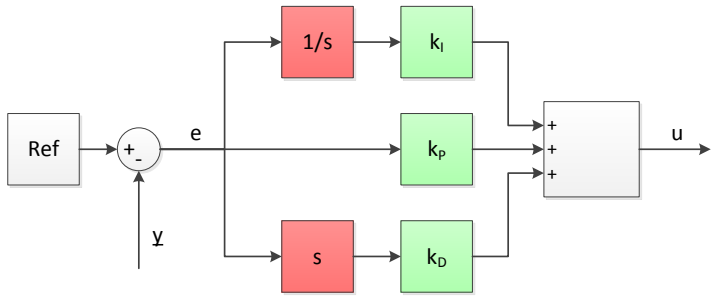


Figure 2.13: Traditional PID structure

be considered with four DoF, attitude and altitude, because, in the $U \rightarrow PWM$ transformation, F_z is needed. In our implementation, the low level hardware acts as a radiocontrolled quadrotor, in which altitude is pilot controlled by a throttle command.

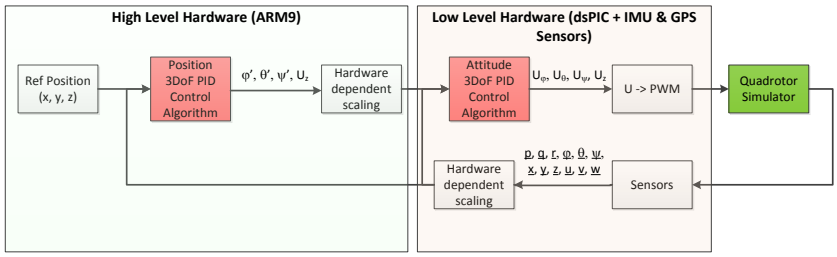


Figure 2.14: Two-layers PID control algorithm

To make accurate simulation of overall system, the simulator needs a block (Hardware dependent scaling) where taking into account electronics delays and sensor dynamics. This may be modelled by a first order filter with a time constant τ_B:

$$u = \frac{1}{\tau_B s + 1} \tag{2.32}$$

Furthermore, a conversion of state variables in integer notation is needed to use DSP embedded functions of dsPIC33 CPU.

2.5.2.1 dsPIC33 fractional type

In the Microchip© development kit for dsPIC33, all present functions are implemented to take advantage of the entire set of embedded DSP instructions. By

this way, the functions are computed using a fractional datatype, starting from the definition of the integer datatype, using the 1.15 format. The notation used to describe a format consists of two numbers separated by a period (.): the first number is the number of bits to the left of radix point, the second is the number of bits to the right of the radix point. For example, 16.0 format is an integer format; all bits lie to the left of the radix point. The 1.15 format is a fractional format with one bit for sign and the remaining ones for the fractional part of the number, so it's possible to represent values within the range $[-1, 1[$.

2.5.3 Roll and pitch channels

Pitch and roll channels have the same transfer functions due to the symmetry of the vehicle. In a linearized approach the transfer function between U_θ (pitch control) and θ (pitch angle) is derived by 2.28 and 2.27 equations:

$$G_\theta = \frac{\theta}{u_\theta} = G'_q \frac{1}{s} = \frac{k_{1y}}{s^2 + k_{2y}s - k_{3y}} \tag{2.33}$$

Where k_{1y} is found previously by a static motor model identification in par. 2.3.3, k_{2y} can be neglected in a first approximation, as it is related to aerodynamic drag, and k_{3y} depends on the distance between the center of gravity and the center of rotation. This term is positive for the constrained quadrotor, but it can be negative or negligible for a free-flying quadrotor. So in the simplest form, the equation 2.33 becomes:

$$G_\theta = \frac{k_{1y}}{s^2} \tag{2.34}$$

Considering the PD controller scheme as in figure 2.15, a simple analytic calibration was made imposing a damping coefficient $\zeta = 0.7$ and a cutoff frequency $\omega_n = 10Hz$ to the closed-loop transfer function.

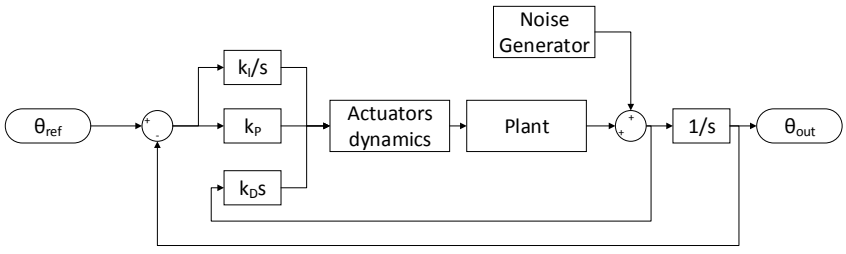
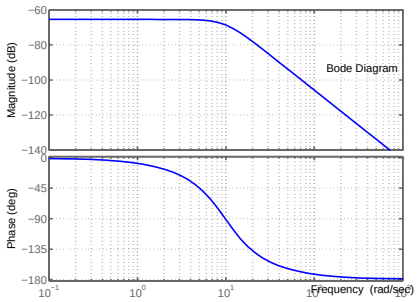
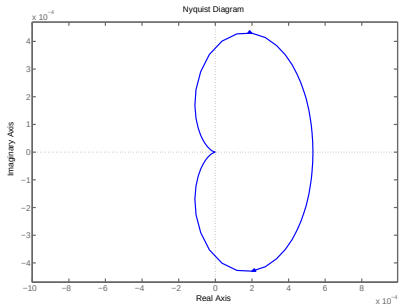


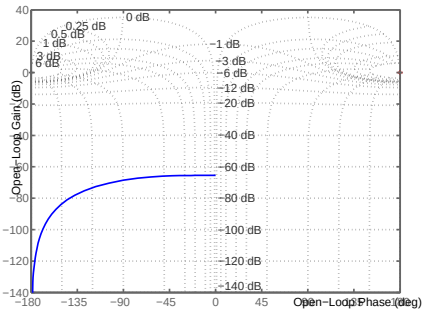
Figure 2.15: PD controller scheme for pitch and roll channels



(a) Bode diagram



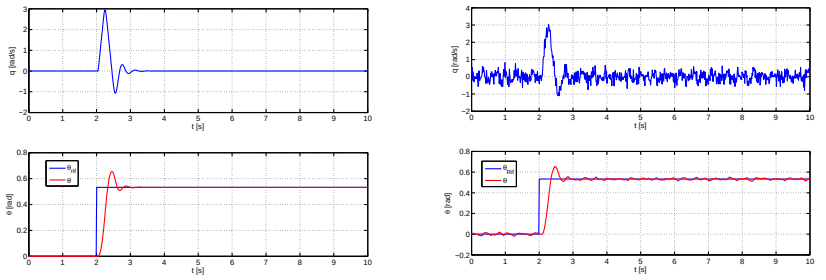
(b) Nyquist diagram



(c) Nichols diagram

Figure 2.16: Diagrams for roll and pitch channel

This first calibration was useful to perform experimental tests on the attitude controller of the constrained quadrotor.



(a) Response Without noise

(b) Response with random white noise

Figure 2.17: System response to step input for pitch channel

However, on the free-flying quadrotor a PID controller is needed to avoid attitude errors in hovering conditions. Considering the PID controller scheme as in figure 2.18, the tuning of the control system was made by an optimization algo-

rithm.

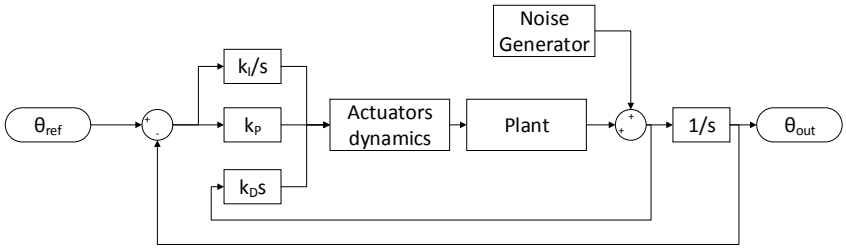


Figure 2.18: PID controller scheme for pitch and roll channels

In particular, a genetic algorithm was configured as in (2.35), to minimize the squared mean error of the plant output.

$$f_{OBJ}(k_P, k_I, k_D) = RMS(\theta - \theta_{ref}) \tag{2.35}$$

The design variables to optimize are the coefficients k_P, k_I, k_D of the PID controller. To guarantee performance and robustness, several constraints on gain margin, phase margin and cutoff frequency were used.

$$\begin{aligned} G_M &> 6dB \\ G_P &> 70deg \\ f_{cutoff} &> 10rad/s \end{aligned} \tag{2.36}$$

2.5.4 Yaw channel

The same procedure can be applied to the $U_\psi - \psi$ yaw transfer function:

$$G_\psi = \frac{\psi}{u_z} = \frac{k_{1z}}{s^2 + k_{2z}s} \tag{2.37}$$

Where k_{1z} depends on motor characteristics (as par. 2.3.3) and k_{2z} on aerodynamical or mechanical (for the constrained quadrotor) damping. However it has to be considered that the friction of the mechanical bearings introduce an effect which has to be neglected for the flying quadrotor. Due to the strong influence of frictional torque of bearing used in the constrained platform for yaw system, this controller was hard to fine tune on the on-ground facility and an additional in-flight tuning was needed.

Neglecting the aerodynamic coefficient, equation 2.37 becomes:

$$G_\psi = \frac{k_1 z}{s^2} \tag{2.38}$$

The low torque coefficient implies a slow response time for this channel. For this reason a PD scheme (as in figure 2.19) was considered for both constrained and free-flying model.

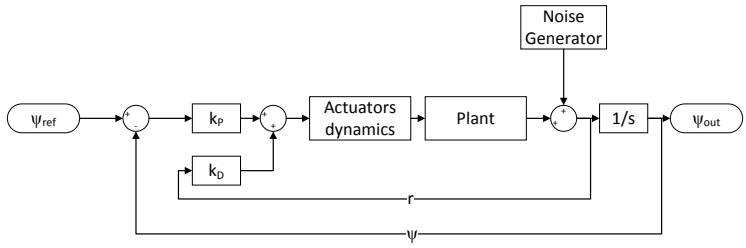
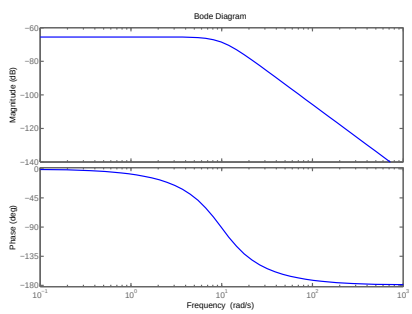
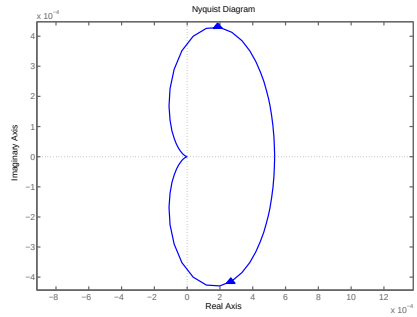


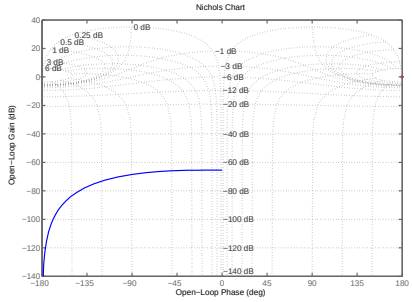
Figure 2.19: PD controller scheme for yaw channel



(a) Bode diagram



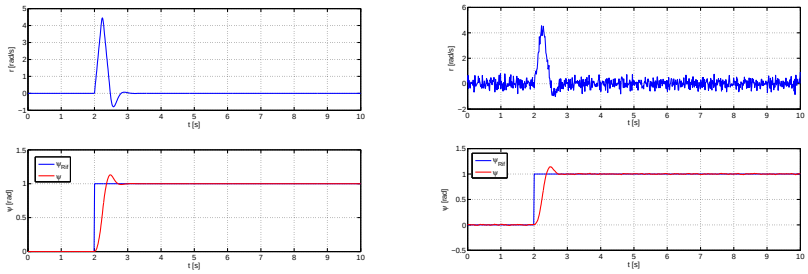
(b) Nyquist diagram



(c) Nichols diagram

Figure 2.20: Diagrams for yaw channel

A simple analytic calibration can be made imposing a damping coefficient $\zeta = 0.7$ and a cutoff frequency $\omega_n = 10Hz$ to the closed-loop transfer function.



(a) Response Without noise (b) Response with random white noise

Figure 2.21: System response to step input for yaw channel

2.6 Autopilot controller

Before introducing the autopilot controller, the attitude control system was implemented on the on-board hardware. This system was tested on the constrained platform and on the free flying quadrotor to check the proper working in a real environment. Furthermore to check model identification the implemented simulator was compared to the real platform using the same input commands and analyzing flight test logs.

The autopilot controller as shown in figure 2.10, is implemented on a different electronic board and it acts passing attitude references to the low level processor.

2.6.1 Altitude channel

To control altitude, an altimeter sensor is needed. In this thesis, a sensor fusion between GPS, barometer and dead reckoning using accelerometers was used. This system can be modelled by a filter with a time constant τ_{sz} as in equation 2.39.

$$S = \frac{1}{1 + s\tau_{sz}} \tag{2.39}$$

The SISO channel, $F_z - z$ can be described by the following transfer function:

$$G_z = \frac{u_z}{z} = \frac{\frac{4k_T}{m}}{s(p_z + s)} \tag{2.40}$$

At zero speed, the system is a double integrator, but aerodynamic drag, due

to the increasing speed, move a pole on the negative part of the real axis. In this case, aerodynamics cannot be neglected, but an in-flight identification is needed to fine tune the controller. Using a simplified aerodynamic model a first PID has been implemented.

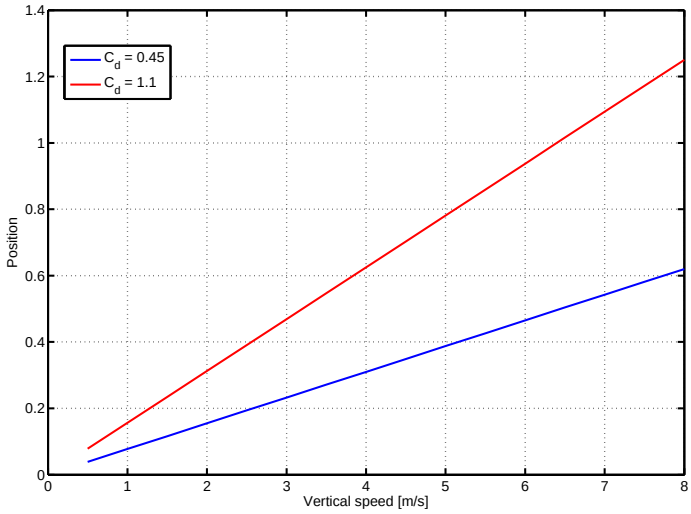


Figure 2.22: Pole position moving due to aerodynamic drag

The cut-off frequency of the transfer function involves in the searching of a slow control system with respect to the attitude controller. A genetic algorithm was configured for an automated tuning of the controller as in (2.41). The objective of the optimization procedure is to minimize the squared mean error between measured altitude and reference.

$$\begin{aligned} f_{OBJ} &= RMS(y - y_{ref}) \\ VP &= \{k_P, k_I, k_D\} \\ G_M &> 6dB \\ G_P &> 70deg \\ f_{cutoff} &> 10rad/s \end{aligned} \tag{2.41}$$

2.6.2 Position controller

Analyzing simulator tests, the system $U_x \rightarrow x$ was modeled as a first order system with $\tau = 0.3$ as time constant. For the trajectory tracking control system a PD controller was implemented (as in figure 2.23 for x axis) and tuned using an optimization procedure, configured similarly to the previous controllers.

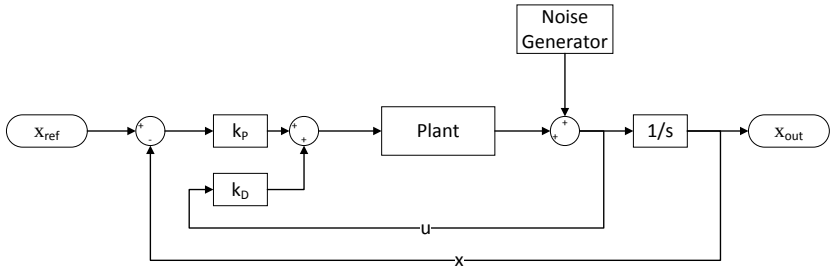


Figure 2.23: PD controller scheme for trajectory tracking

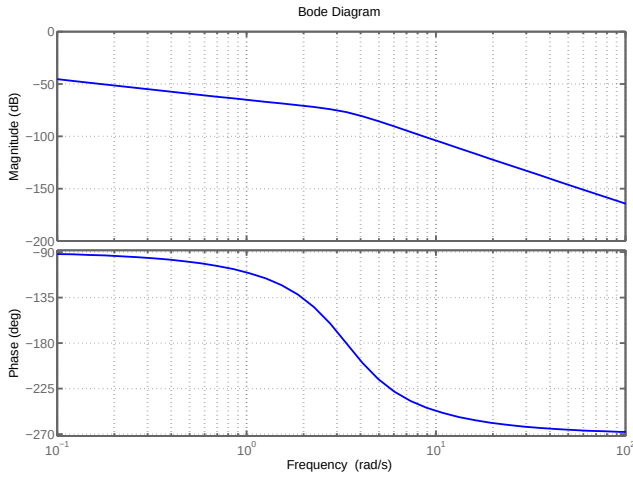


Figure 2.24: Bode diagram for position (x-y) channels

Chapter 3

Optimization

3.1 Introduction

Optimization is the process of making something better. Optimization consists in trying variations on an initial concept and using the information gained to improve on the idea. A computer is the perfect tool for optimization as long as the idea or variable influencing the idea can be input in electronic format.

Optimization algorithms may be divided in two big families:

- traditional optimization methods take the basic approach of heading downhill from an arbitrary starting point then successive improvements increase the speed of the downhill algorithms but do not add to the algorithm ability to find a global minimum instead of a local minimum;

- natural methods generate new points in the search space by applying operators to current points and statistically moving toward more optimal places in the search space. They rely on an intelligent search of a large but finite solution space using statistical methods. The algorithms do not require taking cost function derivatives and can thus deal with discrete variables and discontinuous cost functions. They represent processes in nature that are remarkably successful at optimizing natural phenomena.

In the design of an UAV several optimization problems may be encountered. The first ones are related to the choice of hardware, the design and layout of the structure. Then, during the controller design, an optimization algorithm able to automatically tune the PID controller is useful to automate the tuning procedure

[35, 36].

Finally in the ground guidance system (see 2.3.2) there is the necessity to implement algorithms able to manage the UAV or a fleet of UAVs, automatically optimizing their paths to achieve a target [37].

In this thesis a general purpose optimization software developed during the PhD project is described. This software is based on two well-known natural evolutionary algorithms each one with their own features: genetic and ant colony optimization algorithms. The software is able to solve single and multi-objectives problems.

In particular for multi-objectives implementation two models of solution are proposed: the first based on classic Pareto, the second based on Game Theory. In the following sections, algorithms implementation is described, focusing attention on several original points.

3.2 General Purpose Optimization Software for Engineering Multiobjective Problems

A general multi-objective optimization problem can be described as a vector function f that maps a tuple of m parameters (decision variables) to a tuple of n objectives to maximize (or minimize).

$$\begin{aligned} \max \underline{y} = \max \underline{f}(\underline{x}) = \max(f_1(\underline{x}), f_2(\underline{x}), ..., f_n(\underline{x})) \\ \text{subject to} \quad \underline{x} = (x_1, x_2, ..., x_m) \in X \\ \underline{y} = (y_1, y_2, ..., y_n) \in Y \end{aligned} \tag{3.1}$$

where $\underline{x}$ is called decision vector (and its components decision variables) defined in X, called search space, and $\underline{y}$ is the objective vector (and its components are objective functions) defined in Y, called objective space.

A typical solution to this problem is the research of the Pareto optimal decision vector. Mathematically, considering a maximization problem (as in 3.1), solution

$\underline{a}$ is said to dominate $\underline{b}$ (written as $\underline{a} \succ \underline{b}$) if [38]:

$$\begin{gathered} a, b \in X \\ \forall i \in 1, 2, ..., n : f_i(\underline{a}) \geq f_i(\underline{b}) \bigwedge \\ \exists j \in 1, 2, ..., n : f_j(\underline{a}) > f_j(\underline{b}) \end{gathered} \tag{3.2}$$

In a given set of solutions, all decision vectors which are not dominated by any other one are called *nondominated* solutions. These are denoted as *Pareto optimal* and constitute the *Pareto front*. However, Pareto front is a set of solutions, every one with the same "optimality".

Another concept inherited by game theory branch may be used to solve multiobjective problems.

A *normal form game*

$$\Gamma =< n; X_1, ..., X_n; f_1, ..., f_n > \tag{3.3}$$

is a strategic interaction problem, based on a set of $n \in N$ players, each one choosing simultaneously a strategy in the set of strategies X_i. We denote $X = X_1 \times X_2 \times ... \times X_n$. Player i links each strategy profile $s = (x_1, ..., x_n)$ (solution of the problem) to a profit (or payoff) function $f_i(x) : X \to R$. Every player wants to maximize his profit. In this thesis two solutions to this problem have been used: Nash and Stackelberg equilibria.

3.2.1 Nash equilibrium algorithm

In the Nash equilibrium solution concept no player has anything to gain by changing only his own strategy unilaterally [39].

Considering the game Γ (3.3), a strategy set $(x_1^*, ..., x_n^*) \in X$ is called Nash equilibrium if:

$$\begin{gathered} \forall i = 1, ..., n : \forall x_i \in X_i : \\ f_i(x_1^*, ..., x_n^*) \geq f_i(x_1^*, ..., x_{i-1}^*, x_i, x_{i+1}^*, ..., x_n^*) \end{gathered} \tag{3.4}$$

That is:

$$\max_{x_i \in X} f_i(x_1^*, ..., x_{i-1}^*, x_i, x_{i+1}^*, ..., x_n^*) \tag{3.5}$$

According to this definition, the algorithm for a n players Nash equilibrium

game is presented [40, 41, 42, 43, 44].

The algorithm is based on the *Nash adjustment process* [45], where players take turns setting their outputs, and each player's chosen output is a best response to the output that his opponent chose the period before. If the process does converge, the solution is an optimization of the game.

Let $\underline{x} = \underline{x}_1, ..., \underline{x}_n$ be a feasible solution for the n player Nash problem. Then x_i denotes the subset of variables handled by player i, belonging to a metric space X_i, and optimized by an objective function called f_i. Player i search the optimal solution with respect to his objective function by modifying $\underline{x}_i$.

At each step k of the optimization algorithm, player i optimizes x_i^k using $x_{(-i)}^{k-1} = \underline{x}_1^{k-1}, ..., \underline{x}_{i-1}^{k-1}, \underline{x}_{i+1}^{k-1}, ...\underline{x}_n^{k-1}$. This process exits when all players converge to the same solution. In figure 3.1 the generic Nash co-evolution algorithm for two players is shown.

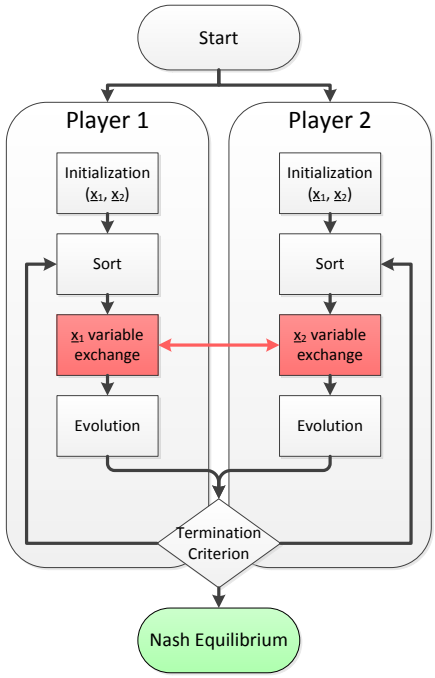


Figure 3.1: Generic Nash co-evolution algorithm

3.2.2 Hierarchical Stackelberg-Nash equilibrium algorithm

In a two-players based Stackelberg game, actors doesn't move simultaneously, but a player, called leader, moves as first, the other one observes and after player 1 moves itself.

Consider a $n+1$ player game, where one player is the leader and the rest of them are followers in a two-level Stackelberg game. Any player is assumed to maximize his own profit called objective function. The followers act in a noncooperative way and play a Nash equilibrium game [39]. The leader takes into account the followers Nash equilibrium, that we assume to be unique, and solve a Nash equilibrium problem in a backward induction scheme. Let $X, Y_1, ..., Y_n$ be compact, nonempty and convex subsets of an Euclidean space that are the leader's and the followers' strategy sets, respectively. Let $l, f_1, ..., f_n$ be real valued functions defined on $X \times Y_1 \times ... \times Y_n$ representing the leader's and the followers' cost functions. We also assume that $l, f_1, ..., f_n$ are continuous in $(x, y_1, ..., y_n)$ and that f_i is strictly convex in y_i for any $i = 1, ..., n$.

For each $x \in X$, that is the leader's decisions, the followers solve the following *lower level Nash equilibrium problem* $N(x)$

$$\begin{cases} \text{find } (\bar{y}_1, ..., \bar{y}_n) \in Y_1 \times ... \times Y_n \text{ such that} \\ f_1(x, \bar{y}_1, ..., \bar{y}_n) = \max_{y_1 \in Y_1} f_1(x, y_1, ..., \bar{y}_n) \\ ... \\ f_n(x, \bar{y}_1, ..., \bar{y}_n) = \max_{y_n \in Y_n} f_1(x, \bar{y}_1, ..., y_n) \end{cases} \tag{3.6}$$

Let $(\tilde{y}_1(x), ..., \tilde{x}_n(x)) \in Y_1 \times ... \times Y_n$ be the unique solution of the problem N(x). The leader has to compute a solution of the following *upper level problem*:

$$\begin{cases} \text{find } \bar{x} \in X \text{such that} \\ l(\bar{x}, \tilde{y}_1(\bar{x}), ..., \tilde{y}_n(\bar{x})) = \max_{x \in X} l(x, \tilde{y}_1(x), ..., \tilde{y}_n(x)) \end{cases} \tag{3.7}$$

Any solution $\bar{x} \in X$ to this problem is called a Stackelberg-Nash strategy, while any vector $(\bar{x}, \tilde{y}_1(x), ..., \tilde{y}_n(x)) \in X \times Y_1 \times ... \times Y_n$ is called a Stackelberg-Nash equilibrium.

The given solution for $n = 1$ is the classical Stackelberg problem solution [39]. This model, for $n > 1$ has been intensively studied and used in different applicative contexts, as in [46, 47, 48, 49].

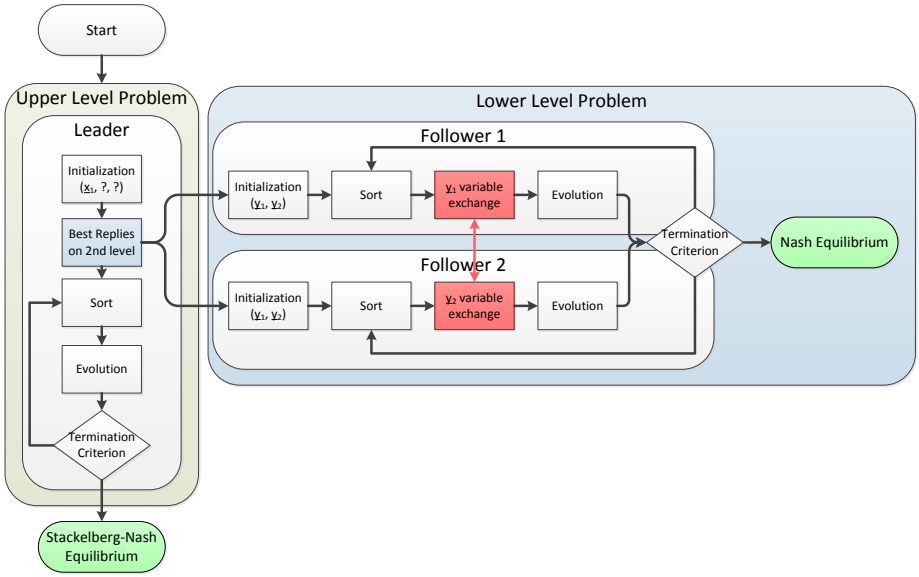


Figure 3.2: Generic Hierarchical Stackelberg-Nash Algorithm

The algorithm is divided in two stages:

- an upper level problem, in which the goal is to maximize the objective function of the leader,
- at each step of the upper level layer, a lower level problem has to be solved, in which the goal is to maximize the objective function of the follower (or find the Nash equilibrium between followers), as the best reply to the leader choice.

3.2.3 Multiobjective Software Implementation

In this thesis the implementation of these methods is organized in two layers via software interfaces:

- IGame: this interface represents the problem to solve. It may be a Stackelberg, Nash equilibrium problem or multiobjective with Pareto front,
- IPlayer: this interface represents the optimization method to use. It may be Genetic or Ant Colony Algorithm. Furthermore, it's possible to use a game

as IPlayer. In this way, a two or more levels Stackelberg-Nash equilibrium can be modelled, but also other hybrid problems may be created.

In the following sections optimization methods are introduced, with several details on the software architecture.

3.3 Genetic Algorithm

3.3.1 Basic concepts

The genetic algorithm (GA) is an optimization and search technique based on the principles of genetics and natural selection. A GA allows a population composed of many individuals to evolve under specified selection rules to a state that maximizes the "fitness" (i.e., minimizes the cost function). The method was developed by John Holland (1975) over the course of the 1960s and 1970s and finally popularized by one of his students, David Goldberg, who was able to solve a difficult problem involving the control of gas-pipeline transmission for his dissertation (Goldberg, 1989). Hollands original work was summarized in his book. He was the first to try to develop a theoretical basis for GAs through his schema theorem. The work of De Jong (1975) showed the usefulness of the GA for function optimization and made the first concerted effort to find optimized GA parameters. Goldberg has probably contributed the most fuel to the GA fire with his successful applications and excellent book (1989). Since then, many versions of evolutionary programming have been tried with varying degrees of success. Some of the advantages of a GA are:

- optimizes with continuous or discrete variables,
- doesn't require derivative information,
- simultaneously searches from a wide sampling of the cost surface,
- deals with a large number of variables,
- is well suited for parallel computers,
- optimizes variables with extremely complex cost surfaces (they can jump out of a local minimum),
- provides a list of optimum variables, not just a single solution,

- may encode the variables so that the optimization is done with the encoded variables,
- works with numerically generated data, experimental data, or analytical functions.

These advantages are intriguing and produce stunning results when traditional optimization approaches fail miserably.

Of course, the GA is not the best way to solve every problem. For instance, the traditional methods have been tuned to quickly find the solution of a well-behaved convex analytical function of only a few variables. For such cases the calculus-based methods outperform the GA, quickly finding the minimum while the GA is still analyzing the costs of the initial population. For these problems the optimizer should use the experience of the past and employ these quick methods. However, many realistic problems do not fall into this category. In addition, for problems that are not overly difficult, other methods may find the solution faster than the GA. The large population of solutions that gives the GA its power is also its bane when it comes to speed on a serial computer the cost function of each of those solutions must be evaluated. However, if a parallel computer is available, each processor can evaluate a separate function at the same time. Thus the GA is optimally suited for such parallel computations.

3.3.2 Binary GA for continuous problems

A genetic algorithm is built on a multi-objective optimization problem, defining a finite population of l chromosomes, several genetic operators, such as crossover and mutation, that move the population to the next state and a termination criterion. The data structure of a GA is principally based on a population of individuals.

Let $X_1, X_2, ..., X_n$ be compact subsets of an Euclidean spaces, denoted as search space. Let $f_1, f_2, ..., f_n$ be real valued functions, defined on $X_1 \times X_2 \times ... \times X_n$, representing the objective functions to be maximized.

Let $s = x_1, x_2, ..., x_n$ be the individual (or chromosome) representing a feasible solution in the search space. As in nature an individual is different from another one thanks to own DNA, in the algorithmic counterpart it's made by a string of bits, called chromosome, that may be divided in several genes, one for each problem variable or property. It's possible to see an individual in two ways: a phenotypic

one, linked to the physic problem and a genotypic one, related to the algorithm.

A finite set of chromosomes make up a population. It can be viewed as a sampling of the problem domain that generation by generation maps zones with an higher probability of presence of the optimum.

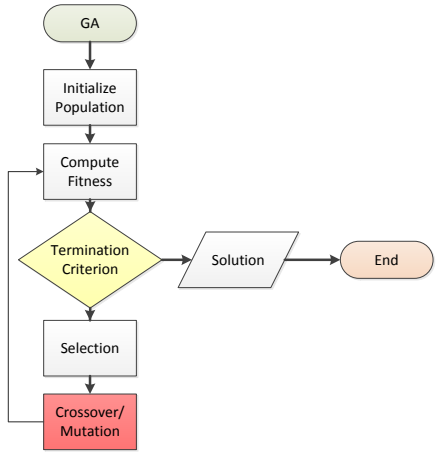


Figure 3.3: Genetic Algorithm flowchart

As shown in figure 3.3, a typical genetic algorithm consists of several steps:

- Population initialization: a set of solutions are randomly generated in the search space. In a binary GA, each chromosome is represented by a fixed-length string of bits, called genotype.

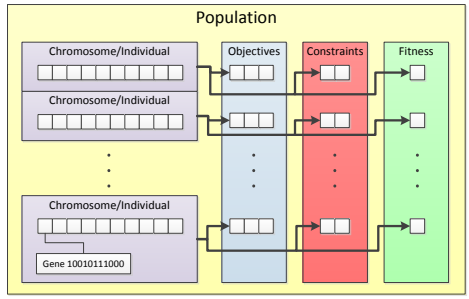


Figure 3.4: Population in a GA

- Fitness computation: at this step function object and constraints are evalu-

ated to sort individuals and get the best solution. In Pareto-front optimizations, chromosomes are sorted by using a ranking method based on domination.

- Termination criterion: usually two criteria are defined in a GA, one on the maximum number of total generations and one on the maximum number of total generations without improvements on the best chromosome.
- Selection: on the sorted population, a probabilistic based selection of parents is made to permit coupling of best individuals without wasting worst chromosomes that may be useful to move towards unexplored zones of search space. Two selection methods have been implemented: roulette wheel and tournament based. In the roulette wheel selection, each chromosome has a probability to be chosen proportional to its fitness (or rank). In the tournament based selection, the same idea is applied on subsets of population in a sort of knockout tournament.
- Crossover: on selected parents, a binary crossover operator is applied to create two new individuals. This operator may be applied in several forms:
  - single-cut: parents chromosomes are mixed using only a single cut;
  - multi-cut: parents chromosomes are mixed using a cut for each gene (property);
  - bit-by-bit: parents chromosomes are mixed considering for each bit the crossover probability. This is the option with most entropy and it's suggested only in few cases.

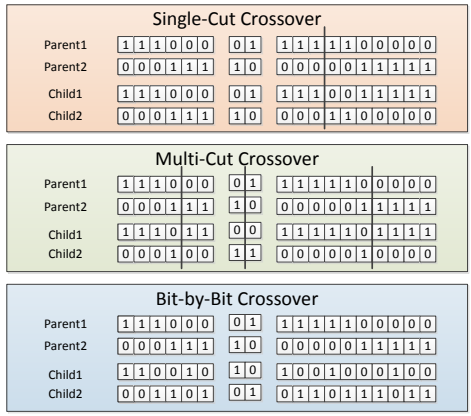


Figure 3.5: Crossover types

- Mutation: to avoid premature stagnation of the algorithm a mutation operator is used, randomly changing a bit of the just created chromosomes. The probability of this operator may be linked to the evolution of the algorithm.

3.4 Ant Colony

3.4.1 Basic concepts

Ants behaviour has always captured the attention of naturalists and ethologists, as they are organized in systems (colonies) that, in despite of the simplicity of constitutive individuals, show a surprising ability to manage very complex tasks.

One of the most astonishing capabilities shown by ant colonies is the dynamic finding for the shortest path between nest and food. Once the food source is found, ants line up in a long, two-way single column, having the ability to dynamically redefine a new best path if a accidental obstacle blocks the actual one.

This efficient colony organization is obtained through the so-called *stigmercy*, that is a form of indirect communication induced by chemical changes applied to the environment. An ant searching for food releases an amount of pheromone on the ground; the perception of this agent by other ants increases the probability that they also follow the same path or a considerable portion of it, but also trying some deviations that could be more effective than the original route.

3.4.2 Ant Colony in graph-based combinatorial problems

A simulated artificial version of this social behaviour is synthesized in the Ant System (AS) metaphor. Ant organization and related interaction rules are replied in a computer system, and the ability to find shortest paths (or equivalent cost functions) in a well-defined environment is exploited in a general way.

Let us consider the minimization of a typical N-P hard combinatorial problem $\Pi(S, f, \Omega)$ (*e.g.* a Generalized Travelling Salesman Problem, or GTSP)[50, 51], where S is the set of solutions, $f(S)$ the objective function and Ω the set of constraints (if any). Aim of the problem is the finding for a feasible, globally optimal solution. This problem is modelled through the following objects: *i*) a finite set $C = [c_1, c_2, ..., c_n]$ of project parameters; *ii*) a complete list of possible states of the system, defined as finite length sequences of elements c_i, $x = \{c_i, c_j, ...c_k, ...\}$.

The collection of the all possible states is denoted with X, and $|X|$ represents the length of X, being the maximum sequence length limited by a positive constant $n \in N$) a set S of candidate solutions, with $S \subseteq X$; *iv*) a set of possible states $\hat{X} \subseteq X$, defined with a problem-dependent function that verifies it is not impossible to complete a sequence $x \in \hat{X}$ in a feasible solution; *v*) a non-void set S^* of optimal solutions with $S^* \subseteq \hat{X} \subset X$; *vi*) a cost function $g(s)$ associated to each candidate solution $s \in S$. Artificial ants will construct solutions making random steps on a complete graph $G_C = (C, L)$ where C represents the set of nodes and L the set of edges. Constraints can be managed in two alternative ways: intrinsically treated at the construction phase (hard way), allowing the ants to built feasible solutions only, or subsequently the construction phase, with infeasible solutions allowed, but treated with penalty functions.

A pheromone trail $\tau_i[\tau_{ij}]$ and a heuristics $\eta_i[\eta_{ij}]$ can be associated to the node c_i [the edge l_{ij}]. The pheromone trail represents a sort of shared memory for the ant colony, spread over the chosen paths, while the heuristics η_i or η_{ij} is an immutable information. In most cases η_i and η_{ij} are strictly connected to the cost function, giving to the ants a valuable support in choosing the best component routes.

The AC algorithm can be seen as a succession of several procedures. Analogously to other natural algorithms, epochs represent the time-base of evolution. For each epoch, ants concurrently build solutions moving themselves on the construction graph, on the basis of the pheromone trails and heuristic information. At each construction step, the ants choose the nodes to switch on, via a probabilistic choice biased on a proportional-random rule:

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N} [\tau_{ij}]^\alpha [\eta_{ij}]^\beta} \tag{3.8}$$

where p_{ij}^k is the probability of transition ij for the k-th ant; τ is the pheromone matrix; η is the heuristic information matrix that depends on the specific problem; exponents α and β are parameters used to bias the influence of pheromone trail, and l is the generic edge not yet visited.

An artificial ant should be equipped with the following abilities/features:

- path exploration in searching for the best solution,
- a general-purpose local memory m^k used for: *i*) partial route storage, *ii*) fea-

sible solutions construction, *iii*) heuristics and objective function evaluation support, *iv*) reverse path reconstruction,

- an initial state x^k and one or more termination rules e^k. Typically, an initial state consists of a void or a single element array.

When set in the state $x_r = \{x_{r-1}, i\}$, the ant should move to a node j of its neighbourhood $N^k(x_r)$ if none of termination rules is verified. When one of termination criteria is meet, the ant stops. This rule may be varied if a different behaviour is preferred for the ant,*i.e.* the construction of infeasible solutions is allowed or not. This is simply obtained modifying heuristics and using properly the ant local memory.

The ant should choose a move using either heuristics and pheromone information previously deposited onto the tracks by the colony.

When a component C_i is added to the current state, the ant should be able to upgrade the pheromone track associated to the component or connection edge.

Once a solution is built, the ant should be able to retrace it in the opposite way, releasing the pheromone on the involved edges.
It should be remarked that all the ants move in parallel and independently of one another except for the pheromone tracking phase, that is performed in a synchronous mode. This circumstance can be seen as a sort of shared learning, where each ant does not adapt itself, but adapt the representation of the problem for the other ones.

3.4.3 Ant System

Colony object is the core of a Ant System algorithm (AS). It contains all the artificial agents, properties and the environment where the ants are allowed to interact, making the stigmercy. Several variants of AS system have been proposed in the literature; the most relevant implementations are: ant-sensity, ant-quantity and ant-cycle, respectively introduced by Dorigo *et al.*[52], Colorni *et al.*[53] and Dorigo [54].

Ant-density and ant-quantity paradigms make a pheromone update after each node-to-node move. On the contrary, ant-cycle performs the update only when

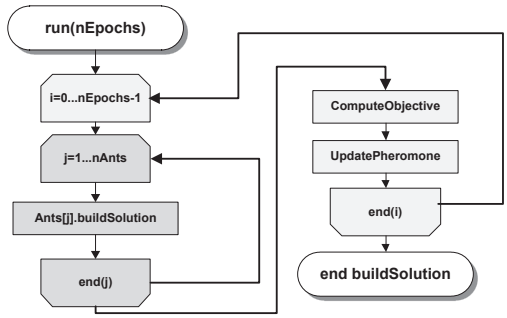


Figure 3.6: Flow-chart of ColonyAS execute function

every ant has completed a loop on the graph; the pheromone quantity released by each ant is proportional to the solution quality *(e.g.* the shortness of the route in a TSP). This variant is the most used because of its superior performance.

The two main stages of an AC algorithm are the solution construction and pheromone update. At the start, the pheromone is slightly overdosed, if compared to the typical amount released by ants during their moves. An evaluation on initial pheromone can be performed with the following expression:

$$\forall(i,j), \tau_{ij} = \tau_0 = \frac{m}{C^{nn}} \tag{3.9}$$

where m and C^{nm} represent respectively the number of ants and the length of circuit generated by employing a nearest-neighbourhood heuristic information only. This initial dosage is made to prevent, on one hand, a premature convergence in initial phases (that would take place with a lower amount), and to avoid, on the other hand, a long preliminary exploration on the graph if a higher amount would be released.

In Fig.3.6, the flow-chart of a ColonyAS with main components is shown. *The ComputeObjective* function is external to the ant cycle i to allow an asynchronous I/O port with external programs and calculus engines.

The pheromone update is performed by the *UpdatePheromone* function that preliminary provides for a reduction of the overall pheromone intensity via an evaporation factor, and subsequently increases the amount on each track proportionally to the quality of the solution expressed by each travelling ant. The evaporation is

controlled by the following expression:

$$\tau_{ij} = (1 - \rho)\tau_{ij} \forall (i, j) \in L \tag{3.10}$$

where $\rho \in [0, 1]$ denotes the pheromone evaporation rate. This parameter should be set to avoid a monotonic growth of the pheromone, allowing the system to forget bad choices and weak solutions.

After the evaporation reduction, ants update the pheromone level according to the following relation:

$$\tau_{ij} = \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \forall (i, j) \in L \tag{3.11}$$

where $\Delta\tau_{ij}^{k}$ is the pheromone amount the ant k releases on the travelled arc ij. This quantity is set as:

$$\Delta\tau_{ij}^{k} = \begin{cases} 1/C^{k} & \text{if } (i, j) \in T^{k} \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

where C^{k} is the length of tour T^{k} assembled by the k-th ant.

3.4.4 Elitist Ant System Colony

The elitist ant system strategy, originally introduced by Dorigo [54], represents a remarkable improvement of the previously described AS. The basic idea is to furnish an increased value, in terms of pheromone track, to the arcs of the best-so-far route on the graph.

This is done by a special ant (the so-called best-so-far ant) that releases an extra amount of pheromone. This behaviour can be seen as example of daemon action working on the ant colony.

The elitist implementation is realised adding to the arcs of the best-so-far tour T^{bsf} a quantity e/C^{bs} , where e is the assigned weight to the best-so-far solution and C^{bs} is its length.

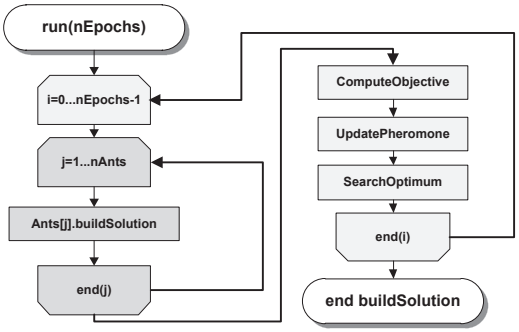


Figure 3.7: Flow-chart of ColonyEAS execute function

Therefore the pheromone upgrade equation becomes:

$$\tau_{ij} = \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} + e\Delta\tau_{ij}^{bsf}\ ,\ \forall(i,j) \in L \tag{3.13}$$

where $\Delta\tau_{ij}^{bsf}$ is the pheromone amount released by the best-so-far ant on the travelled edges:

$$\Delta\tau_{ij}^{bsf} = \begin{cases} 1/C^{bs} & \text{if } (i,j) \in T^{bsf} \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

To avoid a premature convergence, it is preferable to increase the pheromone initially distributed on all the edges:

$$\tau_0 = \frac{e+m}{\rho C^{nm}} \tag{3.15}$$

In Fig.3.7 the general flow-chart of a ColonyEAS is shown with main components.

3.4.5 Colony AS_{RANK}: a rank-based Ant System

A further improvement of the basic AS is the rank-based version AS_{RANK}, proposed by Bullnheimer *et al.*[55]. In the AS_{RANK} metaphor, each artificial ant releases an amount of pheromone proportional to its own rank. Furthermore, as for EAS, the best-so-far ant is also allowed to deposit an extra-amount of pheromone.

At every iteration, only the $(w-1)$ rank-ordered ants, plus the best-so-far one (not necessarily belonging to the actual epoch), are allowed to deposit pheromone.

The best-so-far ant will furnish the greatest feedback, expressed by the weight w, while the rank-based ants will contribute with a rank-weighted amount of pheromone, as expressed by the following expression:

$$\tau_{ij} = \tau_{ij} + \sum_{r=1}^{w-1}(w-r)\Delta\tau_{ij}^{r} + w\Delta\tau_{ij}^{bsf} \tag{3.16}$$

with $\Delta\tau_{ij}^{k} = 1/C^{r}$ and $\Delta\tau_{ij}^{bsf}$. At start, the pheromone is initialized as:

$$\tau_0 = \frac{0.5r(r-1)}{\rho C^{nn}} \tag{3.17}$$

3.4.6 Colony MMAS: a *Max-Min* Ant System

The Max-Min Ant System (MMAS), suggested by Stützle *et al.*[56], introduces several changes to the standard AS:

- the shortest loop found is greatly set off: only the best ant in the actual epoch (or the best-so-far ant) is allowed to deposit pheromone;
- to avoid a premature convergence caused by the extreme elitist strategy implied by the previous statement, a range of achievable pheromone $[\tau_{min}, \tau_{max}]$ values is introduced;
- at start, the pheromone is initialized to τ_{max} and a low rate of evaporation is adopted to favour the domain exploration during the initial epochs;
- every time the algorithm enters in a stagnation phase, or no improvement for the best loop occurs for many consecutive epochs, the pheromone tracks are re-initialized.

Similarly to the ASR, the pheromone update requires a preliminary sorting of the ants with respect to their loop performance (*e.g.* the loop length for TSP). The upgrade is therefore executed according to the following expression:

$$\tau_{ij} = \tau_{ij} + \Delta\tau_{ij}^{best} \tag{3.18}$$

where $\Delta\tau_{ij}^{best}$ is the best ant allowed to deposit pheromone; it can be either the best-so-far ant or the best ant in the epoch (iteration-best ant). The pheromone variation will be set respectively to $\Delta\tau_{ij}^{best} = 1/C^{bsf}$ or $\Delta\tau_{ij}^{best} = 1/C^{ib}$ where C^{ib} is the length of the best loop at the current epoch.

Generally, MMAS implementations alternate both update techniques. The choice of the frequency the two update modes are alternated influences the domain search. When the pheromone upgrade is due to the best-so-far ant, the searching for optimum is quickly biased on the best-so-far loop. On the other hand, an upgrade based on the iteration-best ant, will distribute pheromone in a less biased fashion, as much more connections will receive it.

From testing, as the order of graph increases, an upgrade carried out via the best-so-far ant is more and more required. Therefore, in the present context a feature that considers this eventuality has been implemented. Up to a fifty-node full graph, the probability of choice between the two upgrade modes is set to 50%. This selection is performed extracting a random number k in the range $[1, 350+n]$; if $k \leq 200$, then the iteration-best rule is used; the best-so-far rule otherwise.

According to Stützle *et al.*[57], the pheromone track limits are set to:

$$\begin{aligned} \tau_{max} &= 1/\rho C^{bs} \\ \tau_{min} &= \tau_{max}(1 - \sqrt[n]{0.05})/(avg - 1)\sqrt[n]{0.05} \end{aligned} \tag{3.19}$$

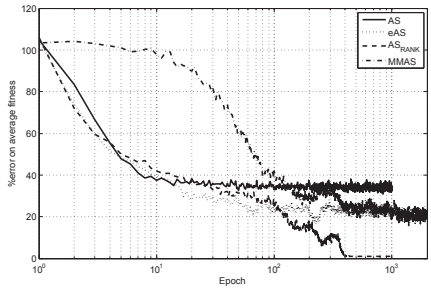
where avg represents the mean of possible choices for each ant at each step of the solution assembly.

In Fig.3.8 A standard TSP test case performed with different ACO (AS, eAS, ASrank and MMAS) is shown for comparison. MMAS algorithm is the best in terms of result quality, but with a slow initial exploration of the graph. ASR algorithm highlights a better convergence speed, but the result shows a slightly reduced quality.

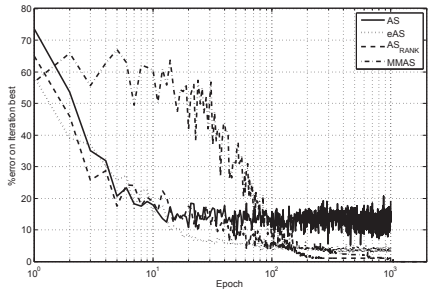
3.5 Parameters Envelope Control System and Hybridization

To increase the flexibility and efficiency of the optimization process, all the previously described algorithms have been slightly modified and predisposed to be dynamically regulated via an Envelope Control System (ECS).

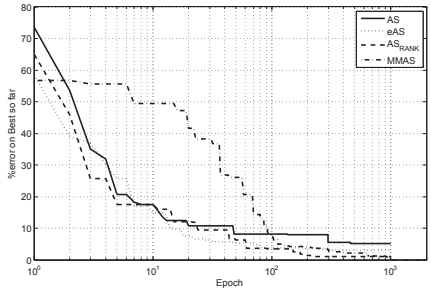
This software structure allows the choice of the working parameters and algorithm type during the simulation. Several control points can be defined by the



(a) Average fitness



(b) Iteration Best



(c) Best so far

Figure 3.8: TSP test case (kroA100); AC algorithms comparison

user on the overall epochs. At each control point, the preferred AC algorithm and the related working parameters can be chosen. A ramped/stepped option is also available to give a better control over continuous/discrete quantities.

A dynamic choice of algorithms and parameters during an AC run can be useful to test more complex strategies that take advantage by the hybridization of methods to obtain an improved efficiency in terms of an higher convergence speed and a more robust domain exploration. To allow the correct working of two (or more) sequential AC algorithms, an implicit parameter exchange is required. Continuity

Table 3.1: Test results on hybrid algorithms

Test	ACO	Epochs	Ants	ρ	α	β	% Error	@Epoch
1	MMAS	50	100	0.02	1	3	1.23	97
	ASrank	950	100	0.2	1	3		
2	ASrank	50	100	0.2	1	3	3.67	48
	MMAS	950	100	0.02	1	3		
3	MMAS	20	100	0.01	1	3	1.00	89
	MMAS	30	100	0.1	1	3		
	ASrank	950	100	0.2	1	3		

is mainly assured by the sharing of the pheromone matrix and the best-so-far ant structure.

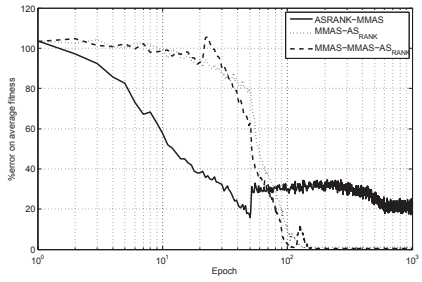
In Fig.3.9 an example of hybridization via envelope controls is shown with a hybrid MMAS-ASR algorithm. In the first test, initial epochs are processed with a Min-Max AS, while remaining iterations are performed with a rank-based AS. In the second test, the sequence of algorithms is inverted. In the third test two MMAS with different evaporation rates plus ASR are performed.

These tests were employed to find the best balance between major features of both methods.

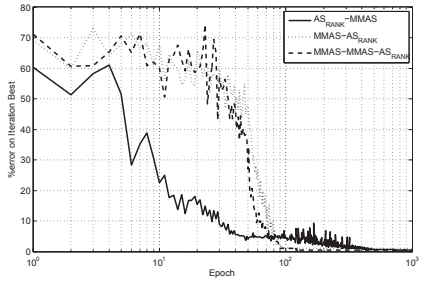
Considering a virtual finishing line at 100 epochs, the results in Tab.3.1 show that the use of MMAS as first method assures a good optimum quality, mainly because of the intrinsic feature of this algorithm to perform a large exploration in initial phases. For this reason, the second test has obviously returned a weak result. The last test shows the importance of parameters also in a hybrid algorithm. A step increment on evaporation further improves the performance of the procedure.

3.6 Distributed Ant Colony

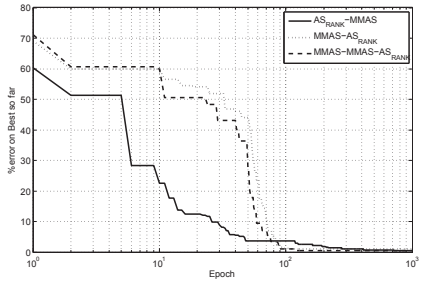
Natural optimization methods as Ant Colony, Genetic Algorithms or Swarm Optimization, produce a large number of candidate solutions at each iteration step (generations or epochs). For each solution, one or more objective functions, if a multi-criterion problem is considered, and one or more state functions, if constrains are present, have to be computed.



(a) Average fitness



(b) Iteration Best



(c) Best so far

Figure 3.9: Hybrid MMAS/AS_{RANK} algorithms comparison

In many research and engineering contexts, as for structural optimization, some of these functions can be very expensive in terms of computational effort.

Two alternative ways can be followed to counteract this problem: *i*) to increase considerably the power of the computer running the overall optimization process; *ii*) to use more computing units, arranged in networks, to perform a distributed optimization task[58].

Generally, first solution only gives a limited improvement rate, as an appreciable increase of elaboration speed requires very expensive high-performance multi-core computers.

On the other hand, a distributed computational layout is very effective because of the intrinsic parallelism of all natural methods. For the GA or ACO concept, each agent (individual or ant) performs its solution independently from each other. Therefore, objective and state functions can be easily evaluated in asynchronous mode on several computers interconnected in a typical master-slave architecture.

Furthermore, many standard up-to-date computers result less expensive than a single high-performance supercomputer, and for distributable algorithms the overall computational power simply increases with the number of computers connected into the network.

Grid and distributed computing has been developed in deep in the last years, and actually many research centers uses large clusters to carry out highly demanding computations. In the present work an effective, low-cost Distributed General Purpose Optimization (DGPO) procedure has been developed to speed-up the overall optimization process.

The proposed DGPO is based on local and remote networks constituted by heterogeneous hardware, made available by cooperative users. This solution allows the reuse of underexploited computers on the network. A well-known similar example is the Seti@Home project, that uses computers offered by volunteers on Internet when not engaged in elaborations (*e.g.* in screensaver mode), to process raw data coming from deep-space scanning radio-telescopes[59].

In Fig. 3.10 a conceptual map of DGPO, illustrating the main features of the network, is shown.

The core of DGPO is the *Remote Method Injection* of the Java Development Kit that allows the distribution on network computers (also on Internet) of some user-defined code able to compute objective and state functions in straight mode, or via external engines (stand-alone commercial codes).

The only mandatory implementation requirement is a network connection and

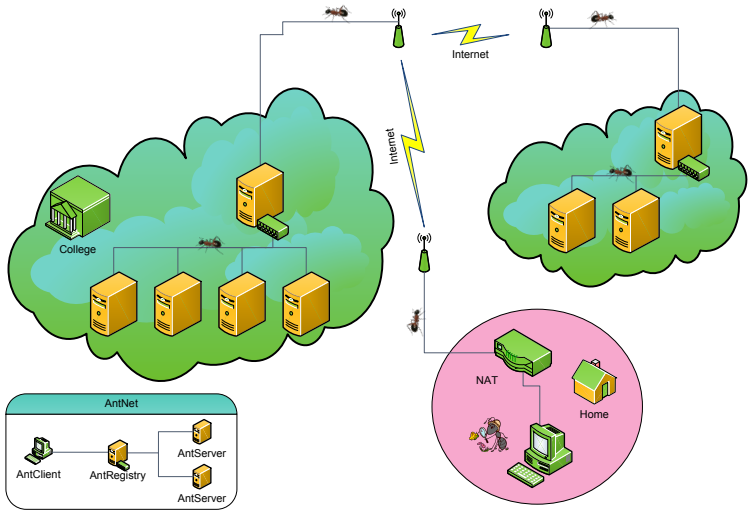


Figure 3.10: DGPO network structure

a small server application enabled on every computer. This distributed method not only allows a virtually unlimited number of connected computers, but it is also extremely versatile as it permits dynamic connection or disconnection of computers during the optimization task.

To obtain these features in the proposed implementation, a non-canonical declaration of server and client roles has been given. Here, the server is just a service furnished to the user; therefore it may be physically remote. The client is the main computer where the Optimization is running. Computers on the network involved in the optimization have been divided in the following categories:

- *Client*: the main computer where ant colony resides;
- *Registry*: an intermediate server that interfaces *Client* with *Server*. It is useful as a log of all associated Servers;
- *Server*: the generic remote computer where objective and state functions are computed.

In a conventional Local Area Network, *Registry* and *Client* may be overlapped, but *Registries* are needed to show the local networks of *Servers* over Internet across the *Network Address Translator* (NAT).

In a heterogeneous grid, a balanced load, according to the processing power of each computer, is a desirable feature. To accomplish it in DGPO, in the early stages of connection with *Registry*, *Servers* are sorted executing a simple benchmark on the processor. In the beginning of the optimization task, *Client* queries *Registries* to create a sorted list of *Servers*. Then each ant sends the solution by choosing the most "powerful" computer available (not running) in the list.

If a previously running *Server* is disconnected or a new *Server* is connected, the related *Registry* notifies it to *Client* that makes a list refresh. If no *Servers* are available, the current ant switch itself on waiting mode, until it is advised of a change. In this way, the distribution of computing load is implicitly balanced, so faster computers on faster network connections will be the mostly used.

Before proceeding with application examples, the computers involved in the grid were preliminarily tested with a performance benchmark. In Tab.3.2, the resulting processing power of each computer present in a local net performing the BURMA-14 test-case (a 14-node full graph TSP) using ACO is shown. Besides the connection type, two main performance indexes are reported: *CPUIndex*, that depends on pure CPU speed (floating point benchmark) and *LinkIndex*, measuring the speed of the specific connection with the *Client* server.

Table 3.2: DGPO Auto-benchmark capability on Burma14 ACO (14-node TSP with dummy cycle)

ID	CPU	Network	CPUIndex	LinkIndex	Time (ms)
PC1	Intel Core 2 E6300	AntClient	72	-	111605
PC2	AMD Turion 64 X2 TL-52	100Mbit/s LAN	68	231	173634
PC3	AMD Sempron 64 2600+	11Mbit/s WiFi	44	43	179408
PC4	AMD Athlon 2400+	ADSL 6Mb/s-512Kb/s	70	2	274078

Preliminary tests performed on a simple TSP (using ACO) have shown a relatively poor efficiency, due to an unbalanced ratio between the objective function cost and the local network speed ("light" objective function versus slow networks). This typically occurs with very simple objective functions, as the CPUs take much less time to compute than to send data over the network.

Table 3.3: Simple DGPO runs on Burma14 ACO (14-node TSP with dummy cycle)

PCs	Threads	Time (ms)	Speed-up
PC1	1	111605	-
PC1+PC2	2	39407	1.40x
PC1+PC2+PC3	3	54915	2.03x
PC1+PC4	2	99848	1.12x

To simulate a "heavier" objective function, a dummy cycle was added to the TSP. In Tab. 3.3 the results of several DGPO runs with different net compositions is shown. First column reports the IDs of computers actually connected for testing, while in columns 3 and 4 the elapsed time and the speed-up factor are respectively reported. In Tab. 3.4 the previous BURMA 14 is finally processed on a unique network layout (formed by PC1 and PC2) with different colony sizes to show the DGPO auto-balancing capability. In the second and third row of the Table 3.4 apparently the same ant colony size is reported. The difference between the two runs is that in the first case a "virtual" 100-ant colony (a 10-ant colony restarted 10 times) is evaluated, while in the latter case a full 100-ant colony is considered. The computational load is automatically redistributed on the basis of PC performance index, as clearly shown in the last two columns.

To obtain a more efficient load balancing, an increased number of ants in the colony is obviously preferred, because they are concurrently sent in exploration.

Table 3.4: DGPO Auto-balancing capability on Burma14 ACO (14-node TSP with dummy cycle)

PCs	Ants	Time (ms)	Variation %	PC1 load %	PC2 load %
PC1+PC2	10	79407	-	55	45
PC1+PC2 (virtual colony)	100*	794070	900.0	55	45
PC1+PC2	100	583259	634,5	65	35

Chapter 4

Optimization applications

The general purpose Optimization software described in the previous chapter has been used in several phases of this project. The main application is based on the trajectory optimization and here the development of several optimization models is presented.

4.1 Trajectory Optimization

Trajectory optimization problems have been widely considered in aerospace engineering fields [60, 61, 62, 37] due to the importance of the defining an optimal flight trajectory that takes into account mission objectives, environment constraints, vehicle capabilities and performance. Mission objectives are usually set in terms of destination and zones to fly over. The constraints can take into account operational limits of the aircraft, as minimum and maximum speed, minimum turning radius, endurance, etc., or the environment in terms of no-fly (or high risk) zones and flight conditions.

The variational formulation is the most rigorous approach. However, since it is difficult to completely solve a real world problem in a closed form by using a variational approach, numerical approaches with direct or indirect methods are often preferred [63, 64, 65, 66, 67, 60, 63, 68, 64, 65, 69, 70]. The latter try to fulfill necessary conditions for optimality resulting from the application of Pontryagin maximum principle, while the former are mainly based on the discretization of the state space and/or of the control variables, and on the translation of the functional

problem into a nonlinear programming problem (NLP) [71, 72].

The widespread use of indirect methods relies on the simplicity of their formulation and on the possibility to easily take into account constraints on state variables and control inputs. On the other side, the use of indirect methods results into more accurate and (at least locally) optimal solutions.

A first rough approach is to generate the trajectory resorting to topological techniques that define a suitable sequence of way points [73, 74, 75, 61]. Nevertheless, whatever sophisticated the interpolation of the way points, the generated trajectories is not guaranteed to be compliant with the aircraft flight envelope. A flight envelope protection system is in charge to control that this holds true [76]. An original approach, employing graph theory and a suitable parameterization of trajectories is proposed in [77]. Probabilistic or potential based methodologies [78, 79, 80] have also been successfully applied to the trajectory generation problem in the field of mobile robotics.

More sophisticated approaches also take into account the vehicle dynamics [81, 82, 83, 84, 85, 86]. Among them, in [83] the trajectory is defined as a sequence of trim conditions and maneuvers (motion primitives), compatibly with vehicle dynamics. The selection of such maneuvers can be done on-line by solving integer programming optimization problems, that however may require a high computational burden.

At the price of a higher computational burden predictive model-based techniques either linear or constrained, can be used [81, 87, 85, 86].

By using a simplified description of the vehicle dynamics, and a reduced set of decision variables, an approach to predictive problems employing both continuous and integer variables can be used. In particular when the objective function and constraints are assumed linear the problem is formulated as a MILP-Mixed Integer Linear Programming. This formulation enables the use of logic expressions modeling decisions in the optimization [72, 88, 89, 90].

The joint presence of both a high number of constraints and of mixed-type variables makes appealing the use of nonconventional optimization techniques, e.g., evolutionary algorithms. Such methods are based on the definition of a population

of potential solutions to the problem, that evolve towards the optimum by using a probabilistic approach. They do not need any information about the function to be optimized, but the possibility to numerically evaluate it, thus overcoming the limitations of the purely deterministic approaches. The research units has a previous experience in the usage of evolutionary techniques in trajectory design problems in the presence of operational and functional constraints associated to the vehicle and the environment [91].

In this thesis numerical methods based on evolutionary optimization algorithm to solve trajectory optimization problems are shown. These problems have the scope of searching efficient flight trajectories to minimize a performance index with specified constraints. During the design phase of ground station for the UAV, several approaches have been considered, using different algorithms and/or combining them.

4.1.1 Spline based trajectory approach

To find feasible trajectories, through a geometrical approach based on topological techniques, a set of n way points may be defined on the scenario space. These points have to be interpolated to generate a trajectory curve such that objective function and constraints may be evaluated. One of the most used interpolating function is the spline, a smooth polynomial function, piecewise-defined:

$$S : [a, b] \rightarrow R \tag{4.1}$$

Where $n-1$ ordered disjoint subintervals $[t_{i-1}, t_i]$ are defined on the interval $[a, b]$:

$$a = t_0 < t_1 < ... < t_{n-2} < t_{n-1} = b \tag{4.2}$$

The spline can be divided in $n-1$ parts, each one for a subinterval:

$$P_i : [t_{i-1}, t_i] \rightarrow R \tag{4.3}$$

Such that:

$$\begin{gathered} S(t) = P_1(t), t_0 \leq t < t_1, \\ S(t) = P_2(t), t_1 \leq t < t_2, \\ ... \\ S(t) = P_{n-1}(t), t_{n-2} \leq t < t_{n-1} \end{gathered} \tag{4.4}$$

Now, consider a and b the starting point and the destination of the aircraft. Let $n-2$ intermediate way points $(\underline{\tilde{x}}_1, ..., \underline{\tilde{x}}_{n-2})$ be the design variables to optimize, a uniform cubic spline curve can be defined on these way points using the parametric form:

$$\begin{aligned} \underline{x}(t) &= a_i + b_i t + c_i t^2 + d_i t^3 \\ \text{such that for } i &= 1, ..., n-2 \\ \underline{x}(t_i) &= \underline{\tilde{x}}_i \end{aligned} \tag{4.5}$$

In the next paragraphs several methods to generate waypoints for the construction of an optimized spline based trajectory are shown and compared, rebuilding all implementation steps from the starting point of this research.

4.1.2 Genetic Algorithm absolute position model

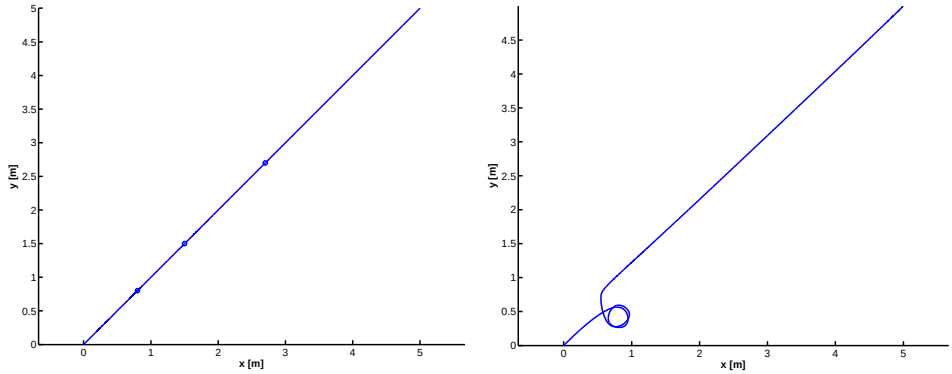
The first step in the research of an algorithm for the optimization of the trajectories of an UAS, was the implementation of a simple objective function to build the spline, starting from the absolute position of a set of waypoints.

Let $A = (x_A, y_A)$ and $B = (x_B, y_B)$ be the position of starting point and objective. The optimization algorithm must look for the minimum path to reach B starting from A. Consider $X \times Y = [x_A, x_B] \times [y_A, y_B]$ the search space, $p_i = (x_i, y_i) \in (X \times Y)$ as the design variables, such that $p_0 = A$ and $p_{end} = B$, and f_{obj} the objective function to minimize:

$$f_{obj} = \int_0^1 S(\underline{p}, t)dt \tag{4.6}$$

Where $S(\underline{p}, t)$ is the parametric spline built on p_i.

This objective function is quite good for the validation of spline construction and the first approach, but the results on a simple scenario without any constraint show (see Figure 4.1) several problems increasing the number of waypoints.



(a) 5 control points

(b) 10 control points

Figure 4.1: GA Absolute position model Scenario 1

4.1.3 Genetic Algorithm relative position model

Waypoints generated in the previous method are not related between themselves, so the optimization algorithm must be spend a lot of time to compute the objective function for unfeasible overlapped trajectories. To overcome this problem, it's needed a way to generate relative waypoints with respect to the trajectory building.

Let $M \times \Phi = [0,1] \times [0,\pi]$ be the search space and $p_i = (m_i, \phi_i) \in (M \times \Phi)$ the design variables. The position of the waypoint i is computed starting from the $(i-1)th$ one such that:

$$\begin{gathered} \underline{x}_i = \underline{x}_{i-1} + m_i \left\{ \begin{array}{c} \cos\alpha_i \\ \sin\alpha_i \end{array} \right\} (\underline{x_{end} - x_0}) \\ \text{with } \alpha_i = \alpha_{min} + \phi_i(\alpha_{max} - \alpha_{min}) \end{gathered} \tag{4.7}$$

Where α_min and α_max are computed starting from the joining line between x_{i-1} and the destination point x_{end} (as in Fig.4.2).

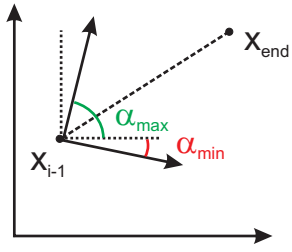


Figure 4.2: Waypoint position computing

Once the functionality of the algorithm has been validated, constraints due to environment and vehicle have been added to check results in presence of complex scenarios.

In particular, constraints on curvature of the trajectory and avoiding obstacles are considered. These constraints are modelled as penalty functions that worse the fitness of unfeasible solutions.

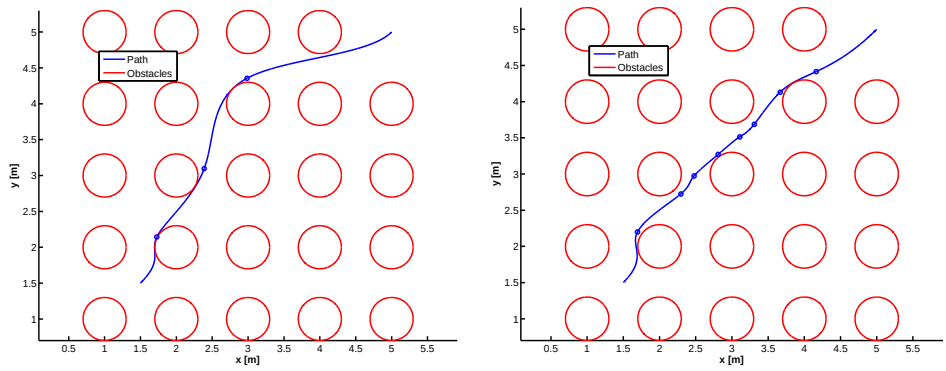
$$f_{obj} = f_{obj}p \left(\frac{g_{max} - g_{min}}{g(x) - g_{min}} \right)^{\gamma} \tag{4.8}$$

Where $g(x)$ is the constraint function, g_{max} and g_{min} are respectively the minimum and maximum allowed value of the function. p is a coefficient to increase the initial weight of penalty function and γ represents an exponent to have an exponential trend away from imposed limits.

A realistic scenario was built with several features:

- take-off point $A = (1.5, 1.5)$;
- landing point $B = (5.0, 5.0)$;
- starting slope $s_1 = \pi/4$;
- ending slope $s_{end} = \pi/4$;
- $length(\underline{p}) = 8$;
- $X \times Y = [0, 5] \times [0, 5]$;
- $\forall i = 1...5 x_{obs} = i, y_{obs} = i$;
- maximum curvature $\ni_{max}= 10$;

In Figure 4.3 results of this model for a complex scenario with several obstacles are shown. Thanks to the change on the search space, the new algorithm shows an improved robustness increasing the number of design variables.



(a) 5 control points

(b) 10 control points

Figure 4.3: GA Relative position model Scenario 2

4.1.4 Ant Colony Regression model

ACO algorithms described in Chapter 3 mainly provide for a great efficiency and robustness (par.3.4.2) in combinatorial and graph-oriented problems. This ability has been conveniently adapted to find the shortest path between two points in a complex scenario with several constraints.

Starting from a topology optimization approach using an Ant Colony algorithm [92], the search space of this optimization consists of a rectangular region defined between take-off point A and a landing point B.

This domain is discretized with quadrangular elements and a boolean matrix, associated to the mesh, keeps track of active elements to reach the destination point.

On this space an ant colony is sent in exploration to find optimal solutions. During its journey, each ant must start from the take-off point, may walk through some defined waypoints and must reach the landing point. The activated elements are used to build a regression spline, useful to smooth the piecewise linear solution.

The first step of the algorithm consists of heuristic matrix and pheromone trail initialization, using an analysis of the full domain and computing a potential field where destination represents an attractive point and obstacles are the rejecting ones. In this way, ants trend is to avoid obstacles and moving towards the destination (see Fig. 4.4).

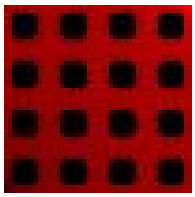


Figure 4.4: Heuristic and pheromone matrix initialization

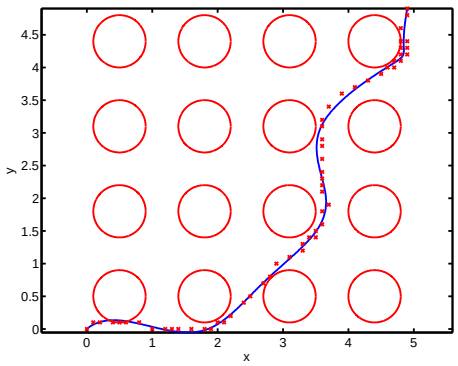
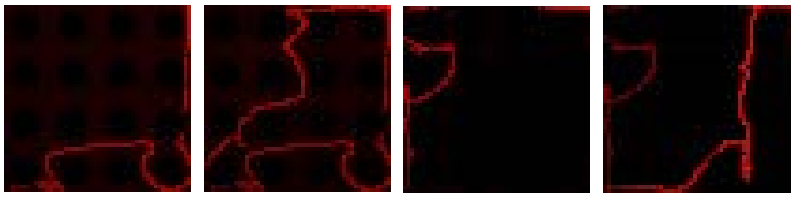
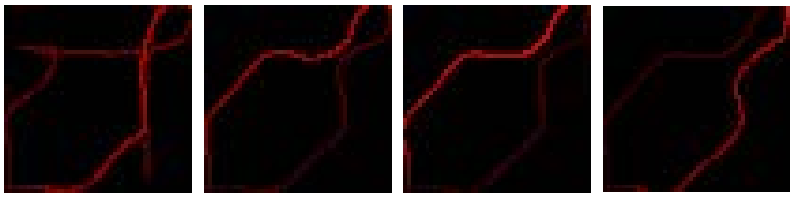


Figure 4.5: ACO Best path

Figure 4.5 shows the best solution for ACO regression model for a complex scenario. As in GA optimization, obstacles and curvature are handled as penalty functions on ant fitness. In Figure 4.6 pheromone matrices in several epochs are shown.



(a) Epoch 2 (b) Epoch 3 (c) Epoch 6 (d) Epoch 14



(e) Epoch 22 (f) Epoch 78 (g) Epoch 118 (h) Epoch 150

Figure 4.6: Pheromone matrix at particular epochs

4.1.5 Ant Colony interpolated spline model

The previous model showed good but sub-optimal solutions and a lack of speed performance and robustness due to the not fixed number of steps for the ant path to reach destination. This behavior is the cause of the generation of very long or not feasible trajectories, at the initial steps of the optimization, making a single ant slow to find the way and releasing pheromone on useless zones of the domain.

To overcome these problems, the ability to skip elements was given to ants. In this model, ants have a fixed number of steps to reach destination and a fixed maximum number of elements on which they can walk without activating the underlying elements.

So the solution have always the same number of active elements, speeding up spline building and increasing algorithm robustness.

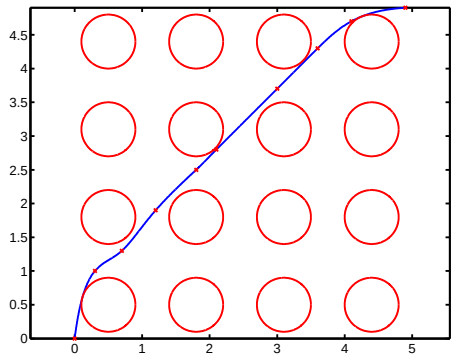
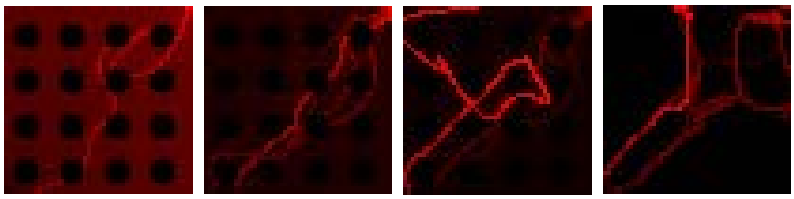
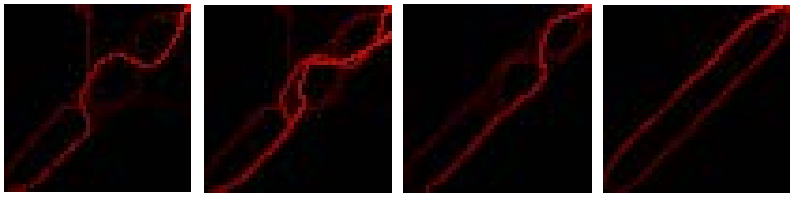


Figure 4.7: ACO Best path

Figure 4.7 shows the best solution for ACO spline model for a complex scenario. As in GA and in ACO regression optimization, obstacles and curvature are handled as penalty functions on ant fitness. In Figure 4.8 pheromone matrices in several epochs are shown. Heuristic and pheromone matrix have been initialized using the same method as in ACO regression model.



(a) Epoch 2 (b) Epoch 7 (c) Epoch 9 (d) Epoch 31



(e) Epoch 39 (f) Epoch 47 (g) Epoch 63 (h) Epoch 111

Figure 4.8: Pheromone matrix at particular epochs

4.1.6 Multiobjective Genetic Algorithm for coverage problems

A surveillance UAV mission planning, to be complete, needs the definition of a coverage area to explore or supervise. In this paragraph, a full scenario is provided with obstacles (no-fly zones) and a coverage area on which the UAV will use a camera to watch it.

The GA relative position model has been modified to deal with one or more coverage area. At the beginning of the optimization phase, a potential field is built taking into account only coverage areas. This works as a modifier in the construction of spline control points.

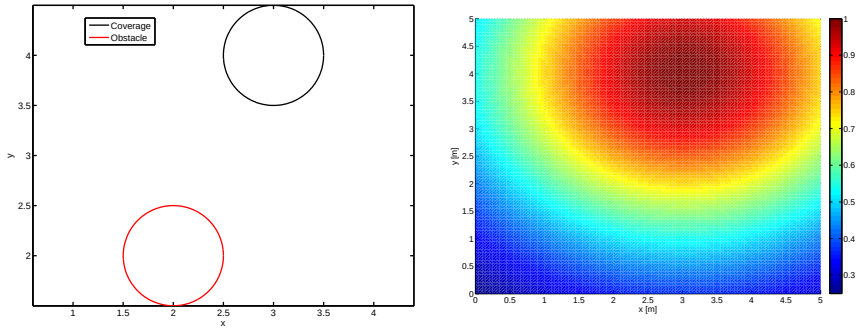
As in 4.1.3, let $M \times \Phi = [0, 1] \times [0, \pi]$ be the search space and $p_i = (m_i, \phi_i) \in (M \times \Phi)$ the design variables. Consider $F(x, y)$ the potential field associated to a circular coverage area such that:

$$F(x, y) = 1/(1 + (r * d(x, y))^2)^2 \tag{4.9}$$

Where $d(x, y) = \sqrt{(x - x_c)^2 + (y - y_c)^2}$ is the distance of the arbitrary point (x, y) from the center (x_c, y_c) of the circular coverage area and r is the radius.

Considering a simple scenario with only one obstacle in $(x_{obs} = 2, y_{obs} = 2)$ and

one coverage area ($x_{cov} = 3, y_{cov} = 4$) as in figure 4.9a, the figure 4.9b shows the potential field.



(a) (b)

Figure 4.9: Scenario and potential field for the first multiobjective example

The position of the waypoint i is computed starting from the $(i-1)th$ one such that:

$$\begin{aligned} \underline{x}_i &= \underline{x}_{i-1} + m_i \left\{ \begin{array}{c} \cos\alpha_i \\ \sin\alpha_i \end{array} \right\} (\underline{x_{end} - x_0}) \\ \text{with } \alpha_i &= \alpha_{min} + \phi_i(\alpha_{max} - \alpha_{min})^{\frac{\Delta_x F}{\Delta_y F}} \end{aligned} \tag{4.10}$$

Where α_{min} and α_{max} are computed starting from the joining line between x_{i-1} and the destination point x_{end}.

Two objective functions are considered, one on the length of trajectory and one on the coverage of the target area:

$$\begin{aligned} f_1(\underline{x}) &= (area_{tot} - area)/area_{tot} \\ f_2(\underline{x}) &= length(S(\underline{x})) \end{aligned} \tag{4.11}$$

Where $area_{tot}$ is the total coverage area, $area$ is the supervised area, taking into account the on-board camera cone of vision.

The following results have been obtained using all multiobjective paradigms of the optimization software described in Chapter 3 for the scenario in figure 4.9.

The first result has been achieved using the scalarization technique and so it's a single-objective optimization, where the two objective functions are combined into a unique fitness.

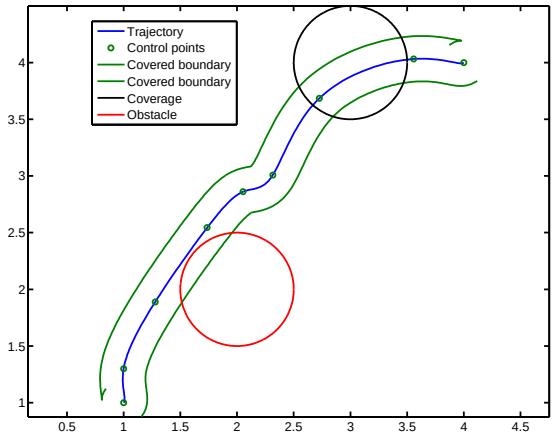
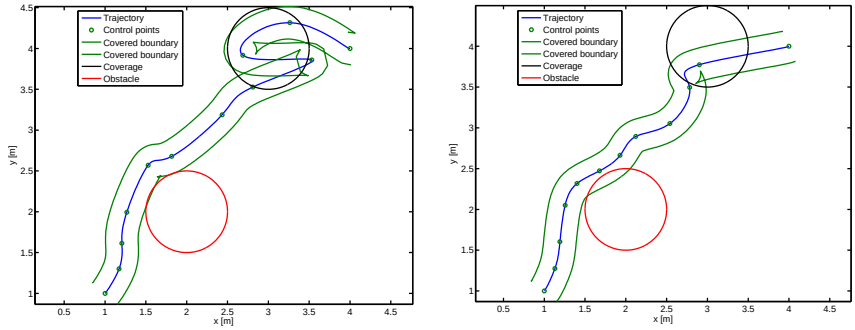


Figure 4.10: Single objective results for Scenario 3

The second result has been obtained using the Pareto front paradigm. In figures 4.11 the two ends of the front are shown.



(a) (b)

Figure 4.11: Multi objective results for Scenario 3

Being unable to divide variables for each player in a Nash Equilibrium game, several tests have been run with different assignments. This is a so-called Virtual Nash Equilibrium Game [93]. In the first instance, all the phases are assigned to *player*1 and all modules to *player*2.

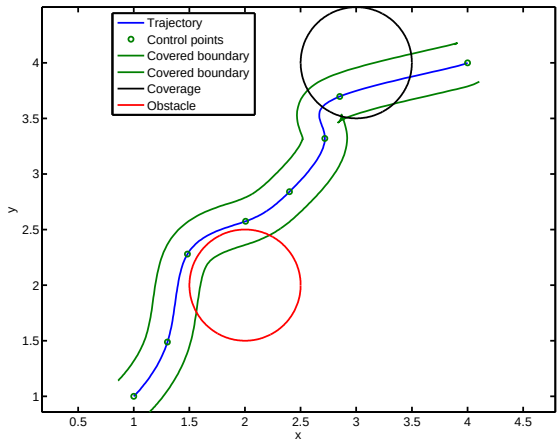


Figure 4.12: Nash Equilibrium results #1 for Scenario 3

The second has been run assigning points alternatively to *player*1 and *player*2.

In the second test:

$$\begin{aligned} p_1 &= (m_1, \phi_1, m_3, \phi_3, m_5, \phi_5) \\ p_2 &= (m_2, \phi_2, m_4, \phi_4, m_6, \phi_6) \end{aligned} \tag{4.12}$$

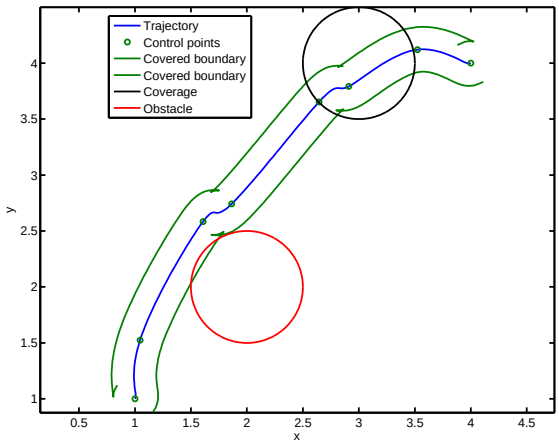


Figure 4.13: Nash Equilibrium results #2 for Scenario 3

The third and fourth tests have been run assigning the first points to one player and the second ones to the other.

In the third test:

$$\begin{aligned} p_1 &= (m_1, \phi_1, m_2, \phi_2, m_3, \phi_3) \\ p_2 &= (m_4, \phi_4, m_5, \phi_5, m_6, \phi_6) \end{aligned} \tag{4.13}$$

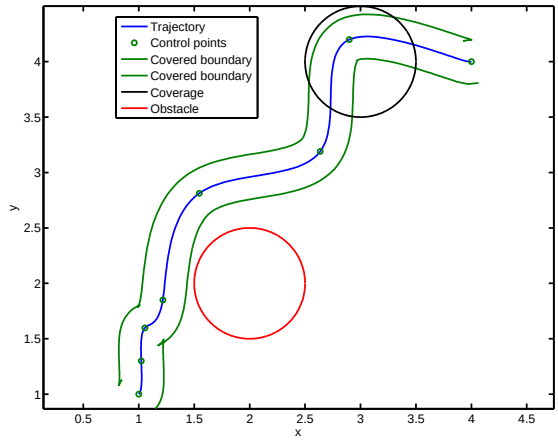


Figure 4.14: Nash Equilibrium results #3 for Scenario 3

In the fourth test:

$$\begin{aligned} p_1 &= (m_4, \phi_4, m_5, \phi_5, m_6, \phi_6) \\ p_2 &= (m_1, \phi_1, m_2, \phi_2, m_3, \phi_3) \end{aligned} \tag{4.14}$$

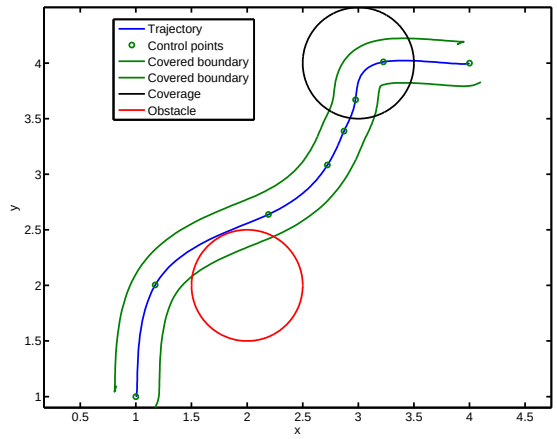


Figure 4.15: Nash Equilibrium results #4 for Scenario 3

In figure 4.18 the objective functions of all results are compared to the pareto

front.

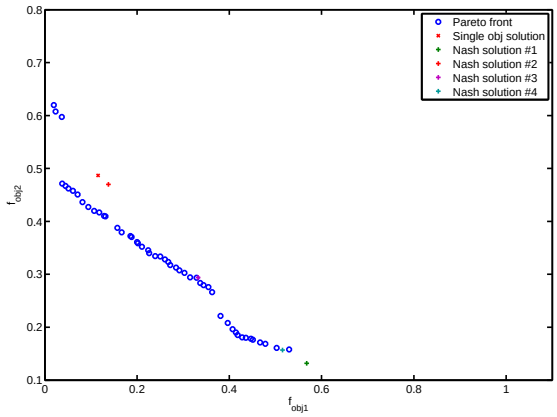
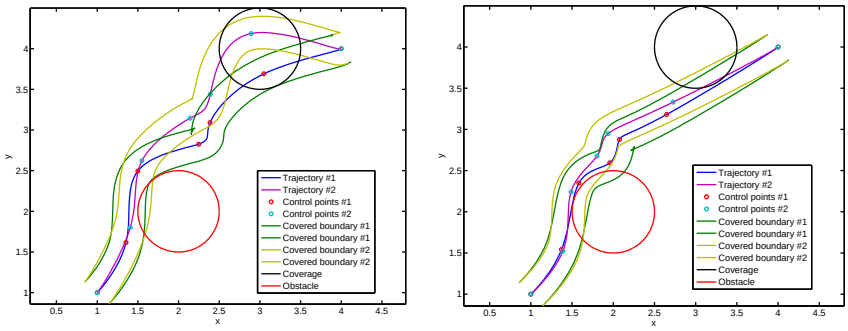


Figure 4.16: Single-objective vs Nash Equilibrium vs Pareto Front

4.1.7 Multi-trajectory Genetic Algorithm for coverage problems

The coverage problem may be achieved using two or more UAV aircrafts. The last step of this trajectory optimization research is to enlarge the fleet of available UAV. In this paragraph only a test case with two aircrafts is shown because the algorithm is still in the development phase. First results are about the multi-objective optimization using pareto front paradigm.



(a) (b)

Figure 4.17: Multi-trajectory, Multi-objective results for Scenario 3

The second result is obtained assigning the control points of trajectory #1 to *player*1 and the control points of trajectory #2 to *player*2.

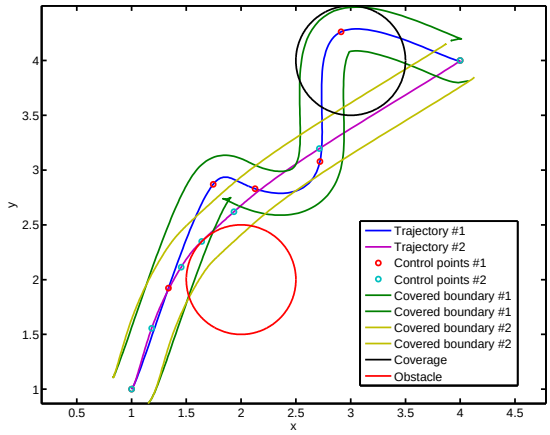


Figure 4.18: Multi-trajectory Nash Equilibrium result for Scenario 3

The algorithm is still in an alpha stage of development, but results are good and must be analysed for a complex scenario with more than two aircrafts.

Chapter 5

Trajectory tracking and Flight tests

In Chapter 2, a quadrotor simulator able to follow a planned path has been developed and thanks to the optimization software built in the ground guidance system (Chapter 3), an optimal trajectory planning has been achieved for an arbitrary scenario with obstacles and constraints on dynamic specifications of the UAV aircraft.

In this chapter, several test cases to validate the coupling of these methodologies are shown. In particular there are results on:

- tests on the low-level control system, using simple reference signals,
- test on the high-level control system, using the planned trajectory for a given scenario.

After these results obtained from the simulator developed in Chapter 2, some flight tests to analyse the electronic implementation of the three DoF control system have been conducted and here are described.

5.1 Tests on three DoF control system

Using a step as reference signal for pitch, the following figures show the results of simulations.

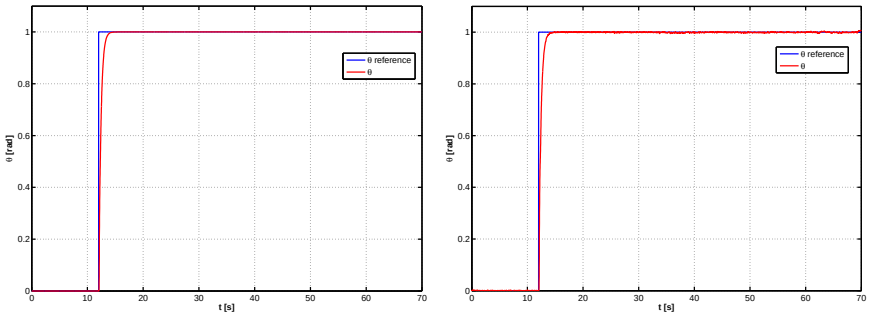


Figure 5.1: Pitch tracking with and without noise

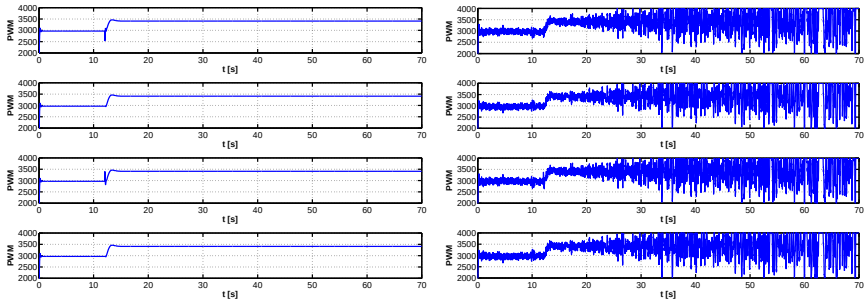


Figure 5.2: PWM signals in pitch tracking with and without noise

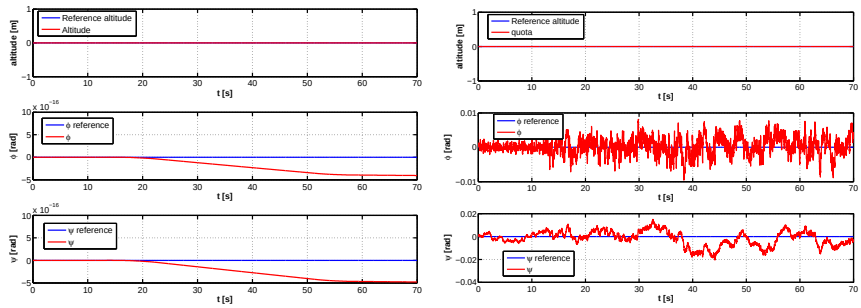


Figure 5.3: Altitude and attitude with and without noise

In figure 5.1 it's possible to note a low response time to follow the reference signal. The control system shows a good robustness to the noise that will be very important in flight tests.

Using a step as reference signal for yaw, the following figures show the results of simulations.

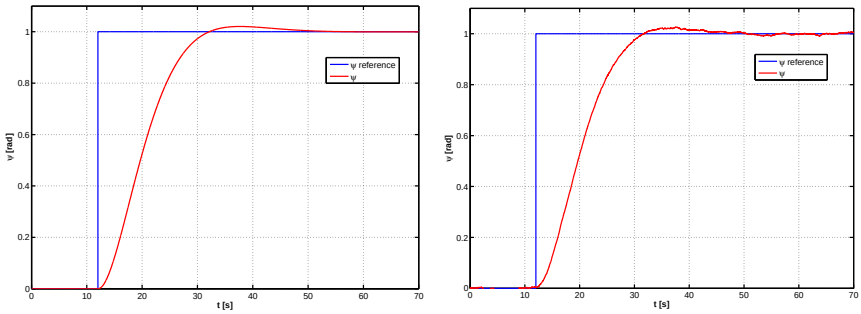


Figure 5.4: Yaw tracking with and without noise

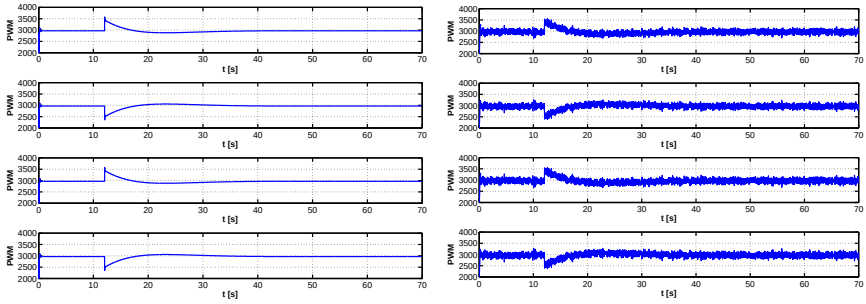


Figure 5.5: PWM signals in yaw tracking with and without noise

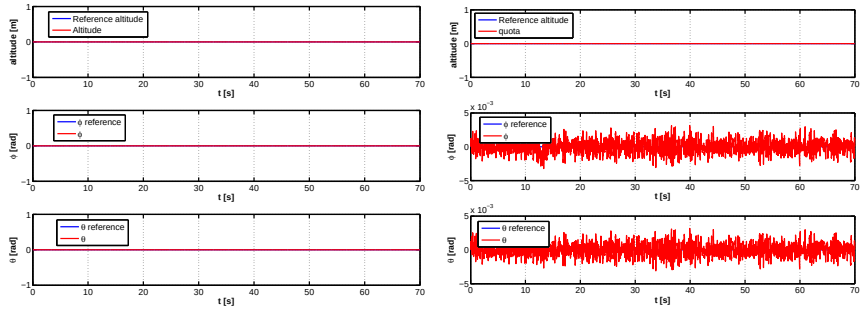


Figure 5.6: Altitude and attitude with and without noise

The response time is slower than other channels because the structure has got more inertia around z axis and the torque coefficient k_R of motors is low, needing more power to obtain a perceptible variation of the yaw angle.

5.2 Flight Tests

In this section the results on the quadrotor flight tests are shown. The on-board telemetry is set to write data at 50 Hz speed. The three DoF control system is tested, using reference signals on pitch and yaw coming from an RC transmitter. This data is used to compare and validate the simulator with the real aerial platform.

In the first flight test two manoeuvres have been performed, one for pitch and one for roll. The first four seconds are needed for the boot phase (for auto-calibrating and auto-testing) of the IMU.

The take-off phase is particularly difficult due to the unsteady aerodynamics and the presence of ground effect. Reached at least an altitude of two meters, the quadrotor can make any maneuver. At $t = 15s$ a pitch command has been given.

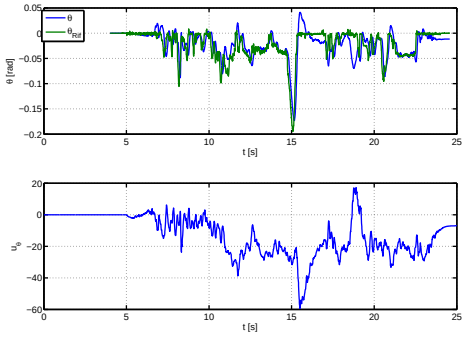


Figure 5.7: Pitch angle, reference and control signal

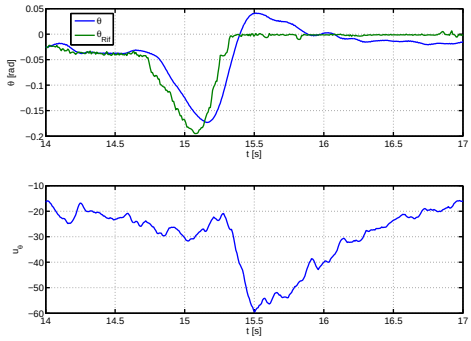


Figure 5.8: Pitch angle, reference and control signal zoom on the manoeuvre

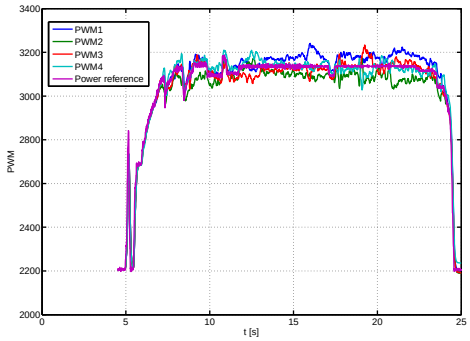


Figure 5.9: PWM signals and power reference

To validate the dynamic simulator, the same input references are used for both quadrotor (real and simulated). In figures 5.10 and 5.12 the comparison is shown, highlighting a quite good identification of the model. Differences are due to the presence of wind and, at the beginning of flight test, in the take-off phase, to the ground effect.

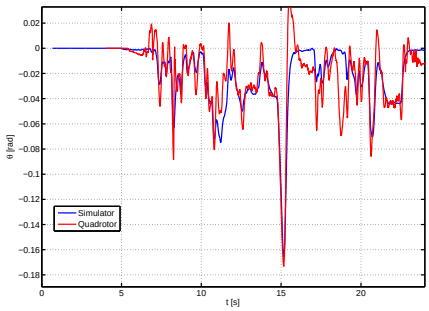


Figure 5.10: Simulator vs Quadrotor pitch angle

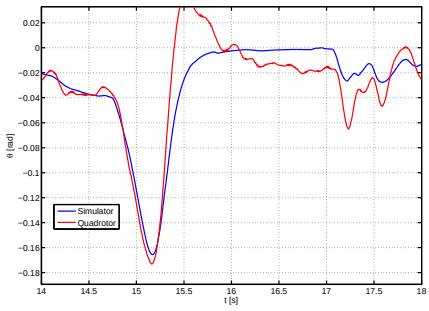


Figure 5.11: Simulator vs Quadrotor pitch angle zoom on the manoeuvre

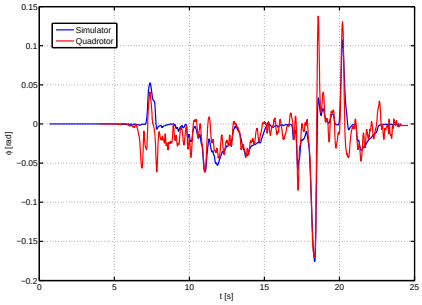


Figure 5.12: Simulator vs Quadrotor roll angle

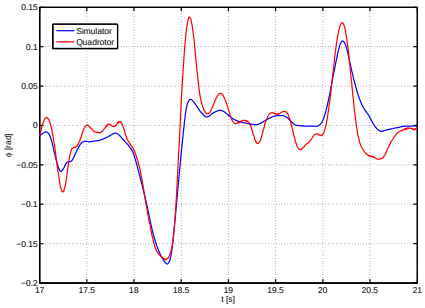


Figure 5.13: Simulator vs Quadrotor roll angle zoom on the manoeuvre

5.3 Tests on six DoF control system

For the trajectory tracking the autopilot control system has been adapted using only reference signals on pitch and roll channels. The yaw angle has been locked at $0deg$. To validate the system, with and without noise, several paths generated in the trajectory optimization phase have been used.

The first test has been conducted on the simplest scenario, without obstacles, to verify the correct working of the autopilot.

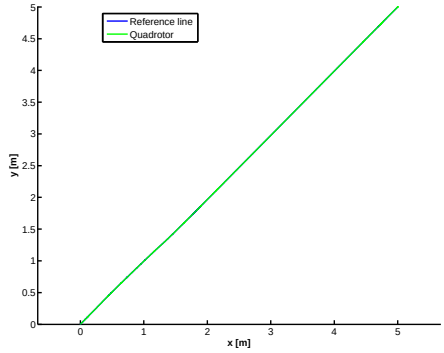


Figure 5.14: Trajectory tracking test #1

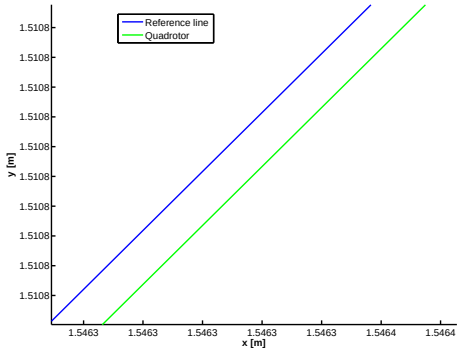


Figure 5.15: Zoom on Trajectory tracking test #1

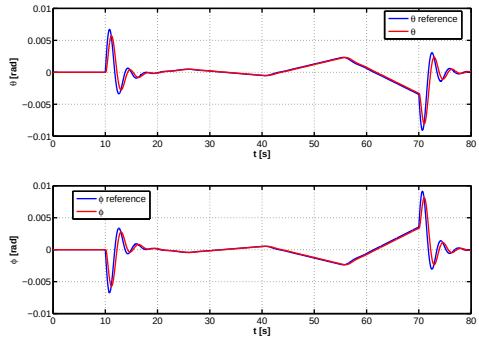


Figure 5.16: Pitch and roll trends

The second test has been conducted on a more complex scenario with two large obstacles. In following figures, trajectory tracking and angle trends are shown.

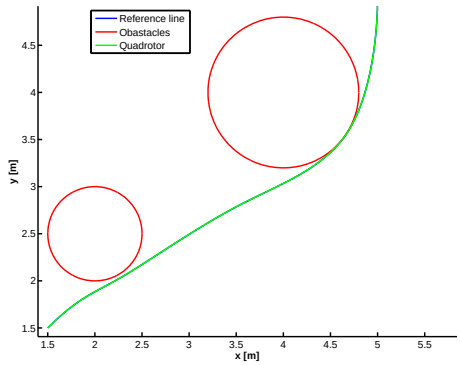


Figure 5.17: Trajectory tracking test #2

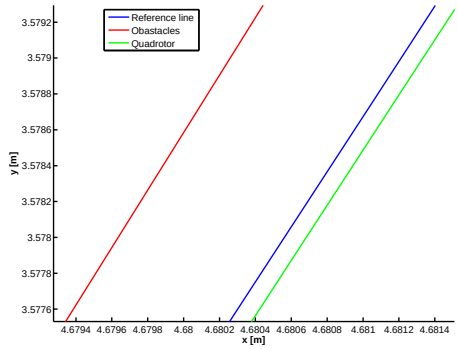


Figure 5.18: Zoom on Trajectory tracking test #2

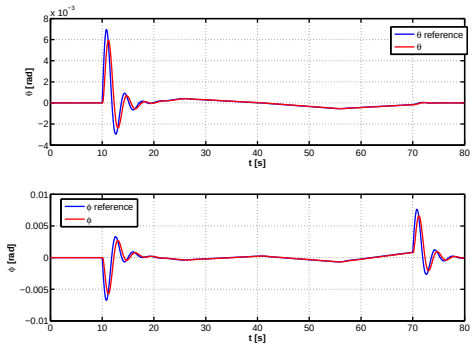


Figure 5.19: Pitch and roll trends

The third test has been conducted on a complex scenario with more obstacles. In following figures, trajectory tracking and angle trends are shown.

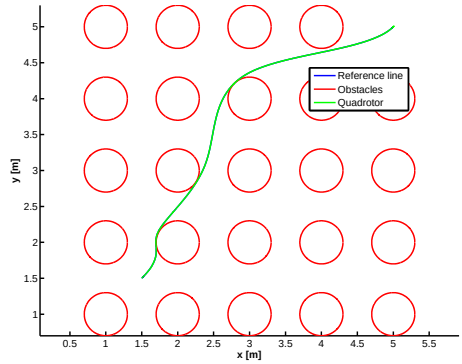


Figure 5.20: Trajectory tracking test #3

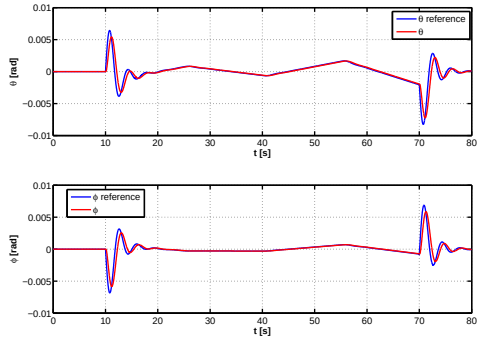


Figure 5.21: Pitch and roll trends

Chapter 6

Conclusions

A prototype of a quadrotor UAV has been built and used to develop and test an automatic navigation and guidance system.

A dynamic simulator has been implemented in Matlab/Simulink to design the control system. Its tuning has been made using optimization algorithms with both SISO and MIMO approaches.

The main scientific contribution of this work is the development of hybrid nature inspired optimization algorithms and their application to the trajectory planning of the UAV.

A general purpose optimization software has been developed and implemented on java platform to be used on a wide range of processors and ready to exploit multithreading capabilities. Furthermore, for large problems, also a grid computing extension to the software has been implemented. This software is based on two largely-known evolutionary algorithms: Genetic algorithm and Ant Colony.

Genetic algorithm is implemented in a classic binary form with several features and options on evolution, crossover and mutation. The Ant Colony is implemented using four different algorithms present in the literature, adding hybridization and dynamic parametrization features.

Another contribution was the implementation of some game theory paradigms (as Nash Equilibrium and Stackelberg Equilibrium) as a software layer over the op-

timization algorithms such that it's possible to combine different paradigms with several algorithms. The validation of this software has not been detailed for the sake of brevity.

In Chapter 4 two possible applications of the developed optimization algorithms has been shown. In particular it was shown how the proposed algorithms can be implemented on the ground station of the UAS to make a trajectory planning which accounts for several environmental constraints, as obstacles and flight mechanics constraints. A multiobjective problem has also been solved, where the aircraft must minimize trajectory length and maximize the supervision of a coverage area in the chosen scenario.

In Chapter 5 the first flight tests for the tuning of the flight control algorithms are shown. The trajectory tracking performance on the flight paths generated with the proposed optimization algorithms are evaluated on the numerical simulator.

Open problems and possible developments originates from this PhD thesis. In particular the mathematical modeling of the UAV requires wind tunnel testing and/or flight experiments, the on board system requires a more appropriate filtering and data fusion to provide a reliable absolute position in the 3D space.

Also the possibility to apply the proposed results to real world problems with acceptable computational burden must be better explored. From this point of view the work done on algorithms and software implementation already allows to think at an efficient implement on GPU architectures and distributed systems.

Bibliography

[1] Barnhart, R. K., E. Shappee, D. M. Marshall, and Others (2011) *Introduction to unmanned aircraft systems*, CRC Press.

[2] Pounds, P., R. Mahony, and P. Corke (2006) "Modelling and control of a quad-rotor robot," in *Proceedings Australasian Conference on Robotics and Automation 2006*, Australian Robotics and Automation Association Inc.

[3] Lee, S., S. H. Kang, and Y. Kim (2011) "Trajectory tracking control of quadrotor UAV," in *Control, Automation and Systems (ICCAS), 2011 11th International Conference on*, IEEE, pp. 281–285.

[4] Bresciani, T. (2008) *Modelling, identification and control of a quadrotor helicopter*, Department of Automatic Control, Lund University.

[5] Gupte, S., P. I. T. Mohandas, and J. M. Conrad (2012) "A survey of quadrotor Unmanned Aerial Vehicles," in *Southeastcon, 2012 Proceedings of IEEE*, IEEE, pp. 1–6.

[6] Amir, M. Y. and V. Abbass (2008) "Modeling of quadrotor helicopter dynamics," in *Smart Manufacturing Application, 2008. ICSMA 2008. International Conference on*, IEEE, pp. 100–105.

[7] Bouabdallah, S. and R. Siegwart (2007) "Full control of a quadrotor," in *Intelligent robots and systems, 2007. IROS 2007. IEEE/RSJ international conference on*, Ieee, pp. 153–158.

[8] Burhans, D. (2007) "A robotics introduction to computer science," in *AAAI spring symposium, March*, pp. 26–28.

[9] Nice, E. B. (2004) *Design of a four rotor hovering vehicle*, Ph.D. thesis, Cornell University.

[10] Escareno, J., S. Salazar-Cruz, and R. Lozano (2006) "Embedded control of a four-rotor UAV," in *American Control Conference, 2006*, IEEE, pp. 6—-pp.

[11] Guenard, N., T. Hamel, and V. Moreau (2005) "Dynamic modeling and intuitive control strategy for an," in *Control and Automation, 2005. ICCA'05. International Conference on*, vol. 1, IEEE, pp. 141–146.

[12] Bouabdallah, S., P. Murrieri, and R. Siegwart (2005) "Towards autonomous indoor micro VTOL," *Autonomous Robots*, **18**(2), pp. 171–183.

[13] Hoffmann, G., D. G. Rajnarayan, S. L. Waslander, D. Dostal, J. S. Jang, and C. J. Tomlin (2004) "The Stanford testbed of autonomous rotorcraft for multi agent control (STARMAC)," in *Digital Avionics Systems Conference, 2004. DASC 04. The 23rd*, vol. 2, IEEE, pp. 12—-E.

[14] Bouabdallah, S., P. Murrieri, and R. Siegwart (2004) "Design and control of an indoor micro quadrotor," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 5, IEEE, pp. 4393–4398.

[15] Castillo, P., A. Dzul, and R. Lozano (2004) "Real-time stabilization and tracking of a four-rotor mini rotorcraft," *Control Systems Technology, IEEE Transactions on*, **12**(4), pp. 510–516.

[16] Castillo, P., R. Lozano, and A. Dzul (2005) "Stabilization of a mini rotorcraft with four rotors," *IEEE control systems magazine*, **25**(6), pp. 45–55.

[17] Salazar-Cruz, S., A. Palomino, and R. Lozano (2005) "Trajectory tracking for a four rotor mini-aircraft," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, IEEE, pp. 2505–2510.

[18] Tayebi, A. and S. McGilvray (2004) "Attitude stabilization of a four-rotor aerial robot," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 2, IEEE, pp. 1216–1221.

[19] ——— (2006) "Attitude stabilization of a VTOL quadrotor aircraft," *Control Systems Technology, IEEE Transactions on*, **14**(3), pp. 562–571.

[20] Andrievsky, B., A. Fradkov, and D. Peaucelle (2005) "Adaptive control experiments for LAAS Helicopter benchmark," in *2005 International Conference on Physics and Control, PhysCon 2005*, pp. 760–765.

[21] Morel, Y. and A. Leonessa (2006) "Direct adaptive tracking control of quadrotor aerial vehicles," in *Proc. of the 2006 Florida Conference on Recent Advances in Robotics*, pp. 1–6.

[22] Madani, T. and A. Benallegue (2006) "Backstepping sliding mode control applied to a miniature quadrotor flying robot," in *IEEE Industrial Electronics, IECON 2006-32nd Annual Conference on*, IEEE, pp. 700–705.

[23] ——— (2006) "Control of a quadrotor mini-helicopter via full state backstepping technique," in *Decision and Control, 2006 45th IEEE Conference on*, IEEE, pp. 1515–1520.

[24] Coza, C. and C. J. B. Macnab (2006) "A new robust adaptive-fuzzy control method applied to quadrotor helicopter stabilization," in *Fuzzy Information Processing Society, 2006. NAFIPS 2006. Annual meeting of the North American*, IEEE, pp. 454–458.

[25] Abeywardena, D. M. W., L. A. K. Amaratunga, S. A. A. Shakoor, and S. R. Munasinghe (2009) "A velocity feedback fuzzy logic controller for stable hovering of a quad rotor UAV," in *Industrial and Information Systems (ICIIS), 2009 International Conference on*, IEEE, pp. 558–562.

[26] Dunfied, J., M. Tarbouchi, and G. Labonte (2004) "Neural network based control of a four rotor helicopter," in *Industrial Technology, 2004. IEEE ICIT'04. 2004 IEEE International Conference on*, vol. 3, IEEE, pp. 1543–1548.

[27] Waslander, S. L., G. M. Hoffmann, J. S. Jang, and C. J. Tomlin (2005) "Multi-agent quadrotor testbed control design: Integral sliding mode vs. reinforcement learning," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, IEEE, pp. 3712–3717.

[28] Mahony, R., S. H. Cha, and T. Hamel (2006) "A coupled estimation and control analysis for attitude stabilisation of mini aerial vehicles," in *Proceedings of the 19th Australiasian Conference on Robotics and Automation*.

[29] Mahony, R., T. Hamel, and J. M. Pflimlin (2008) "Nonlinear complementary filters on the special orthogonal group," *Automatic Control, IEEE Transactions on*, **53**(5), pp. 1203–1218.

[30] Premerlani, W. and P. Bizard (2009) "Direction cosine matrix imu: Theory," *gentlenav. googlecode. com/files/DCMDraft2. pdf, No Date Given, Viewed*, **26**.

[31] Lou, L., X. Xu, J. Cao, Z. Chen, and Y. Xu (2011) "Sensor fusion-based attitude estimation using low-cost MEMS-IMU for mobile robot navigation," in *Information Technology and Artificial Intelligence Conference (ITAIC), 2011 6th IEEE Joint International*, vol. 2, IEEE, pp. 465–468.

[32] Li, J. and Y. Li (2011) "Dynamic analysis and PID control for a quadrotor," in *Mechatronics and Automation (ICMA), 2011 International Conference on*, IEEE, pp. 573–578.

[33] Sa, I. and P. Corke (2012) "System identification, estimation and control for a cost effective open-source quadcopter," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, pp. 2202–2209.

[34] Ljung, L. (1999) *System identification*, Wiley Online Library.

[35] Fantinutto, R., G. Guglieri, and F. B. Quagliotti (2005) "Flight control system design and optimisation with a genetic algorithm," *Aerospace Science and Technology*, **9**(1), pp. 73–80.

[36] Minisci, E. A., G. Avanzini, S. D'Angelo, and M. Dutto (2008) "Multi-objective design of robust flight control systems," .

[37] Guglieri, G. (2008) "Optimal trajectory tracking for an autonomous Uav," *Automatic Control in Aerospace*, **1**, pp. 1–9.

[38] Zitzler, E. and L. Thiele (1999) "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, **3**(4), pp. 257–271.
URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=797969'escapeXml='false'/>`

[39] Basar, T., G. J. Olsder, G. J. Clsder, and T. Baser (1995) *Dynamic noncooperative game theory*, vol. 200, SIAM.

[40] D'Amato, E., E. Daniele, L. Mallozzi, and G. Petrone (2012) "Equilibrium strategies via GA to stackelberg games under multiple follower's best reply," *International Journal of Intelligent Systems*, **27**(2), pp. 74–85.

[41] DAmato, E., E. Daniele, L. Mallozzi, G. Petrone, and S. Tancredi (2012) "A Hierarchical MultiModal Hybrid Stackelberg–Nash GA for a Leader with Multiple Followers Game," *Dynamics of Information Systems: Mathematical Foundations*, pp. 267–280.

[42] D'Amato, E., E. Daniele, L. Mallozzi, and G. Petrone (2011) "Multilevel N Leader–M Follower Decision Making Models with Genetic Algorithms and Applications," *GAME THEORY AND MANAGEMENT. Collected abstracts of papers presented on the Fifth International Conference Game Theory and Management/Editors Leon A. Petrosyan and Nikolay A. Zenkevich.–SPb.: Graduate School of Management SPbU, 2011.–268 p. The collectio*, p. 66.

[43] ——— (2011) "N leader - M follower coalition games with genetic algorithms and applications," in *Proceedings of Eurogen 2011 Evolutionary and Deterministic Methods for Design, Optimization and Control*, pp. 852–866.

[44] ——— (2012) "Waiting time costs in a bilevel location-allocation problem," in *Contributions to game theory and management, Vol.5*, S.Petersburg, pp. 178–186.

[45] Fudenberg, D. and J. Tirole (1991), "Game theory. 1991," .

[46] Luo, Z. Q., J. S. Pang, and D. Ralph (1996) *Mathematical programs with equilibrium constraints*, Cambridge University Press.

[47] Mallozzi, L. and J. Morgan (2006) "On approximate mixed Nash equilibria and average marginal functions for two-stage three-players games," *Optimization with Multivalued Mappings*, pp. 97–107.

[48] Marcotte, P. and M. Blain (1991) "A Stackelberg-Nash model for the design of deregulated transit system," *Dynamic Games in Economic Analysis*, pp. 21–28.

[49] Sherali, H. D., A. L. Soyster, and F. H. Murphy (1983) "Stackelberg-Nash-Cournot equilibria: characterizations and computations," *Operations Research*, **31**(2), pp. 253–276.

[50] Dorigo, M. and L. M. Gambardella (1997) "Ant colonies for the travelling salesman problem," *Biosystems*, **43**(2), pp. 73–81.
URL http://linkinghub.elsevier.com/retrieve/pii/S0303264797017085

[51] ——— (1997) "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, **1**(1), pp. 53–66.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=585892

[52] Dorigo, M., V. Maniezzo, and A. Colorni (1991) *Positive feedback as a search strategy*, *Tech. Rep. June*, Dipartimento di Elettronica e Informatica, Politecnico di Milano, IT.
URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.52.6342

[53] Colorni, A., M. Dorigo, and V. Maniezzo (1991) "Distributed Optimization by Ant Colonies," in *Compute* (F. J. Varela and P. Bourgine, eds.), vol. 142, Elsevier Publishing, pp. 134–142.
URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Distributed+Optimization+by+Ant+Colonies#0

[54] Dorigo, M. (1992) *Optimization, Learning and Natural Algorithms*, Ph.D. thesis, Dip. Elettronica, Politecnico di Milano.
URL http://ci.nii.ac.jp/naid/10016599043/

[55] Bullnheimer, B., R. F. Hartl, and C. Strauss (1997) "A new rank-based version of the ant system: a computational study," *Central European Journal for Operations Research and Economics*, **7**(1), pp. 25–38.
URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.49.4735

[56] Stützle, T. and H. H. Hoos (1999) "MAX-MIN Ant System and Local Search for Combinatorial Optimization Problems." in *MetaHeuristics Advances and Trends in Local Search Paradigms for Optimization* (I. H. O. S Voss S Martello and C. Roucairol, eds.), Boston,MA:Kluwer Academics, pp. 313–329.
URL http://en.scientificcommons.org/42980723

[57] Stutzle, T. and H. Hoos (1997) "MAX-MIN Ant System and local search for the traveling salesman problem," in *Proceedings of 1997 IEEE International Conference on Evolutionary Computation ICEC 97* (Z. M. Piscataway T Bäck and X. Yao, eds.), vol. 16, Ieee, pp. 309–314.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=592327

[58] Mackie, I. (2007) "Object oriented implementation of distributed finite element analysis in .NET," *Advances in Engineering Software*, **38**(11-12), pp. 726–737.
URL `http://hdl.handle.net/10588/1697`

[59] Young, E. and P. Cliff (2001) "Distributed Computing the SETI@home project," *Ariadne*, (27).
URL `http://www.ariadne.ac.uk/issue27/seti/`

[60] Bryson Jr, A. E. (1996) "Optimal control-1950 to 1985," *Control Systems, IEEE*, **16**(3), pp. 26–33.

[61] Yang, H. and Y. Zhao (2004) "Trajectory Planning for Autonomous Aerospace Vehicles amid Known Obstacles and Conflicts," *Journal of Guidance Control and Dynamics*, **27**(6), pp. 997–1008.

[62] Bouktir, Y., M. Haddad, and T. Chettibi (2008) "Trajectory planning for a quadrotor helicopter," in *Control and Automation, 2008 16th Mediterranean Conference on*, IEEE, pp. 1258–1263.

[63] Betts, J. T., N. Biehn, S. L. Campbell, and W. P. Huffman (2000) "Compensating for order variation in mesh refinement for direct transcription methods," *Journal of computational and applied mathematics*, **125**(1), pp. 147–158.

[64] Pesch, H. J. (1989) "Real-time computation of feedback controls for constrained optimal control problems. part 2: A correction method based on multiple shooting," *Optimal Control Applications and Methods*, **10**(2), pp. 147–171.

[65] von Stryk, O. and R. Bulirsch (1992) "Direct and indirect methods for trajectory optimization," *Annals of Operations Research*, **37**(1), pp. 357–373.

[66] Hargraves, C. R. and S. W. Paris (1987) "Direct trajectory optimization using nonlinear programming and collocation." *Journal of Guidance, Control and Dynamics*, **10**(4), pp. 338–342.

[67] Subchan, S. and R. Zbikowski (2009) *Computational Optimal Control*, Wiley Online Library.

[68] Binder, T., L. Blank, W. Dahmen, and W. Marquardt (2000) "Grid refinement in multiscale dynamic optimization," *Computer Aided Chemical Engineering*, **8**, pp. 31–36.

[69] Geiger, B. R., J. F. HAM, G. L. Sinsley, J. A. Ross, L. N. Long, and A. F. Niessner (2008) "Flight Testing a Real-Time Direct Collocation Path Planner," *Journal of guidance, control, and dynamics*, **31**(6), pp. 1575–1586.

[70] Laurent-Varin, J., M. Haddou, and C. Talbot (2007) "Interior-Point Approach to Trajectory Optimization," *Journal of Guidance Control and Dynamics*, **30**(5), pp. 1228–1238.

[71] Betts, J. T. (1998) "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance Control and Dynamics*, **21**(2), pp. 193–207.

[72] Borrelli, F., D. Subramanian, A. U. Raghunathan, and L. T. Biegler (2006) "MILP and NLP techniques for centralized trajectory planning of multiple unmanned air vehicles," in *American Control Conference, 2006*, IEEE, pp. 6—-pp.

[73] Hwang, Y. K. and N. Ahuja (1992) "A potential field approach to path planning," *IEEE Transactions on Robotics and Automation*, **8**(1), pp. 23–32.

[74] Erzberger, H. and H. Q. Lee (1971) "Optimum horizontal guidance techniques for aircraft," *Journal of Aircraft*, **8**(2), pp. 95–101.

[75] Asseo, S. J. (1998) "In-flight replanning of penetration routes to avoid threat zones of circular shapes," in *Aerospace and Electronics IEEE National Conference*, pp. 383–391.

[76] Yavrucuk, I., S. Unnikrishnan, and J. V. R. Prasad (2003) "Envelope protection in autonomous unmanned aerial vehicles," in *ANNUAL FORUM PROCEEDINGS-AMERICAN HELICOPTER SOCIETY*, vol. 59, Citeseer, pp. 2000–2010.

[77] Mattei, M. and L. Blasi (2010) "Smooth flight trajectory planning in the presence of no-fly zones and obstacles," *Journal of guidance, control, and dynamics*, **33**(2), pp. 454–462.

[78] Khatib, O. (1985) "Real-time obstacle avoidance for manipulators and mobile robots," in *International Conference on Robotics and Automation*.

[79] Cen, Y., L. Wang, and H. Zhang (2007) "Real-time Obstacle Avoidance Strategy for Mobile Robot Based On Improved Coordinating Potential Field with Genetic Algorithm," in *IEEE Conference on Control Applications*, pp. 415–419.

[80] Gavrilova, M., J. Cortes, and R. Jarvis (2008) "Computational Geometry in Navigation and Path Planning [From The Guest Editors]," *IEEE Robotics & Automation Magazine*, **15**(2), pp. 6–7.
URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4539713`

[81] Anderson, E. P., R. W. Beard, and T. W. McLain (2005) "Real-time dynamic trajectory smoothing for unmanned air vehicles," *Control Systems Technology, IEEE Transactions on*, **13**(3), pp. 471–477.

[82] Capozzi, B. J. (2001) *Evolution-based path planning and management for autonomous vehicles*, Ph.D. thesis, University of Washington.

[83] Frazzoli, E., M. A. Dahleh, and E. Feron (2002) "Real-time motion planning for agile autonomous vehicles," *Journal of Guidance Control and Dynamics*, **25**(1), pp. 116–129.

[84] Schouwenaars, T., B. Mettler, and E. Feron (2004) "Hybrid Model for Trajectory Planning of Agile Autonomous Vehicles," *Journal of Aerospace Computing Information and Communication*, **1**(12), pp. 629–651.

[85] Schouwenaars, T., J. How, and E. Feron (2004) "Receding horizon path planning with implicit safety guarantees," in *American Control Conference, 2004. Proceedings of the 2004*, vol. 6, IEEE, pp. 5576–5581.

[86] Singh, L. and J. Fuller (2001) "Trajectory generation for a UAV in urban terrain, using nonlinear MPC," in *American Control Conference, 2001. Proceedings of the 2001*, vol. 3, IEEE, pp. 2301–2308.

[87] Dever, C., B. Mettler, E. Feron, J. Popovic, and M. McConley (2006) "Nonlinear trajectory generation for autonomous vehicles via parameterized maneuver classes," *Journal of guidance, control, and dynamics*, **29**(2), pp. 289–302.

[88] Valenti, M., T. Schouwenaars, Y. Kuwata, E. Feron, J. How, and J. Paunicka (2004) "Implementation of a manned vehicle-UAV mission system," in *AIAA Guidance, Navigation, and Control Conference and Exhibit, Providence*, pp. 16–19.

[89] Richards, A. and J. P. How (2002) "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 3, IEEE, pp. 1936–1941.

[90] Schouwenaars, T., B. De Moor, E. Feron, and J. How (2001) "Mixed integer programming for multi-vehicle path planning," in *European Control Conference*, Citeseer, pp. 2603–2608.

[91] Blasi, L., S. Barbato, and M. Mattei (2012) "A particle swarm approach for flight path optimization in a constrained environment," *Aerospace Science and Technology*.

[92] D'Amato, E., L. Iuspa, and G. Del Core (2011) "A distributed ant colony algorithm for topological optimization of 2D structural domains," in *Proceedings of Eurogen 2011 Evolutionary and Deterministic Methods for Design, Optimization and Control* (C. Poloni, D. Quagliarella, J. Périaux, N. Gauger, and K. Giannakoglou, eds.), pp. 108–123.

[93] Tang, Z., J.-A. Désidéri, and J. Périaux (2007) "Multicriterion Aerodynamic Shape Design Optimization and Inverse Problems Using Control Theory and Nash Games," *Journal of Optimization Theory and Applications*, **135**(3), pp. 599–622.
URL `http://www.springerlink.com/index/10.1007/s10957-007-9255-4`