UNIVERSITÀ DEGLI STUDI DI PADOVA

—

DEPARTMENT OF INDUSTRIAL ENGINEERING

Dipartimento di Ingegneria Industriale

—

MASTER'S DEGREE IN MECHANICAL ENGINEERING

TRAJECTORY TRACKING CONTROL OF AN AERIAL MANIPULATOR IN PRESENCE OF DISTURBANCES AND MODEL UNCERTAINTIES

Supervisor: prof. Silvio Cocuzza

Author: Mattia Pedrocco

Student ID: 2020172

ACADEMIC YEAR 2022-2023

To my family...

“ Imagination is more important than knowledge. For knowledge is limited, whereas imagination embraces the entire world, stimulating progress, giving birth to evolution ”

Albert Einstein

Contents

Abstract

The objective of this thesis is to present and test in a virtual environment, kinematic control methods for trajectory tracking by UAMs (Unmanned Aerial Manipulators or aerial manipulators). These consist of UAVs (Unmanned Aerial Vehicles), typically multicopters, equipped with one or more robotic arms that give the system manipulation capabilities. Compared to that of a traditional fixed-base manipulator, the controller of a UAM, in order to make the end organ follow the desired trajectory, must also take into account the displacements of the aerial platform that may be caused by unpredictable external agents or by the movement of the manipulator itself.

After an introductory part on manipulators and UAVs, the physical and mathematical modeling of free-base multibody systems, such as UAMs, is discussed. The equations developed here form the basis of a virtual simulation environment, implemented in MATLAB code, that will be used to perform tests on the control of aerial manipulators. In this environment, are also modeled some of the most common sources of disturbance for this type of systems.

Two algorithms for kinematic control of UAMs are then presented: one at the velocity level, called by the acronym RMRC (Resolved Motion Rate Control), and one at the acceleration level, called by the acronym RAC (Resolved Acceleration Control). The concept of control at the acceleration level is new to aerial manipulators. Therefore, this thesis is concluded by presenting the results of several virtual experiments conducted in the MATLAB simulator to validate the RAC algorithm.

Abstract in lingua italiana

L'obiettivo di questa tesi é quello di presentare e testare in ambiente virtuale, metodi di controllo cinematico per l'inseguimento di traiettorie da parte di UAMs (Unmanned Aerial Manipulators o manipolatori aerei). Questi consistono in UAVs (Unmanned Aerial Vehicles), tipicamente multicotteri, dotati di uno o più bracci robotici che conferiscono al sistema capacità di manipolazione. Rispetto a quello di un manipolatore tradizionale a base fissa, il controllore di un UAM, per far seguire all'organo terminale la traiettoria desiderata, deve tenere conto anche degli spostamenti della piattaforma aerea che possono essere causati da agenti esterni imprevedibili o dal movimento stesso del manipolatore.

Dopo una parte introduttiva su manipolatori e UAVs, viene trattata la modellazione fisico-matematica dei sistemi multibody a base libera, quali sono gli UAMs. Le equazioni qui sviluppate sono alla base di un ambiente di simulazione virtuale, implementato in codice MATLAB, che servirà per effettuare test sul controllo di manipolatori aerei. In quest'ambiente sono anche modellate alcune delle più comuni sorgenti di disturbo per questo tipo sistemi.

Si presentano quindi due algoritmi per il controllo cinematico degli UAMs: uno al livello delle velocità, chiamato con l'acronimo di RMRC (Resolved Motion Rate Control), ed uno al livello delle accelerazioni, chiamato con l'acronimo RAC (Resolved Acceleration Control). Il concetto di controllo al livello delle accelerazioni é nuovo per manipolatori aerei. Per questo si conclude la tesi presentando i risultati di diversi esperimenti virtuali condotti nel simulatore MATLAB per la validazione dell'algoritmo RAC.

Chapter 1

Aerial manipulation

In recent years, a significant growth in Unmanned Aerial Vehicle (UAV) industry has been realized. To this date, UAVs have been used for a large variety of tasks such as remote sensing of agricultural products, forest fire monitoring, search and rescue, border monitoring, transmission line inspection, plant assets inspection and other tasks in which UAVs provided to be useful observation tools. Nowadays the robotics community aims to use UAVs for more active missions than simple inspections, surveillance or monitoring. Researchers have begun examining the possibilities of UAVs to interact with the environment in order to accomplish tedious tasks that are too dangerous or simply too repetitive for human workers. For this scope these should be able to perform operations such as perching, grasping and manipulation that are typical of human hands and that nowadays can be led also by robotic manipulators. The general idea is therefore to integrate one or more of such robotic manipulators into UAV platforms in order to create dexterous "flying arms". The research field that studies this integration is called *aerial manipulation* and falls within the broader well-studied research category known as mobile manipulation.

Hence, by definition, for *aerial manipulation* it is intended the grasping, transporting, positioning, assembly and disassembly of mechanical parts, measurements instruments and any objects, performed with UAVs. Machines that are capable of the before mentioned tasks are generally called aerial manipulators. They are indicated with the acronym UAMs (Unmanned Aerial Manipulators) throughout this work. UAMs contains therefore two main physical subsystems: a UAV platform and a manipulation mechanism, with the necessary sensors and control systems for its autonomous or semi-autonomous functionality (see Fig. 1.1). UAV platform is discussed in more detail in Chapter 2, manipulators in Chapter 3 while UAM sensory configurations is quickly described in 2.4.1.

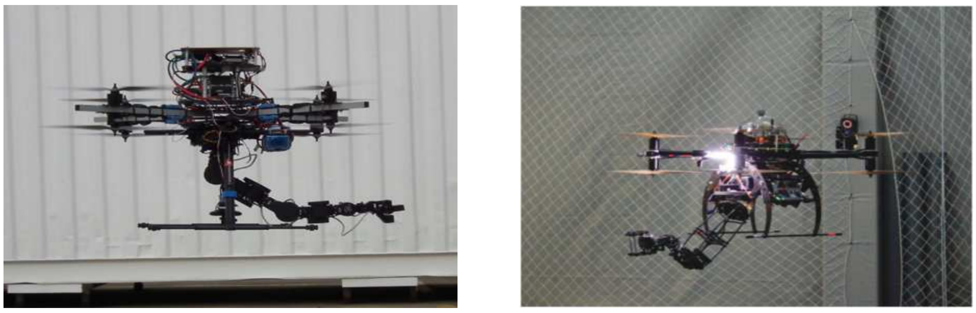


Figure 1.1: Examples of UAMs ([1], [2])

This chapter instead is intended to give the reader a general idea of what are the possibilities and the challenges related to the relatively new research area that is aerial manipulation. More can be found in [1],[3],[2],[4].

1.1 Aerial manipulation missions and scenarios

Missions

Up to now most of aerial manipulation missions types fell into three classes.

Most of UAMs have been used for load transportation purposes. Loads have been moved either by hanging it below the UAV with tethers or grabbed using a gripper or a manipulator. All these methods are constrained by the limited payload capability of UAMs. While the first method alleviates the need for additional mechanism attached to the aircraft and thus allows to carry an heavier load, the second and, in particular, the third method are preferred for the possibility to automatize the pick up process and for the stability that they can guarantee.

A second important class of aerial manipulation missions are those that require force/torque exertion to the environment. In this class, a wide range of

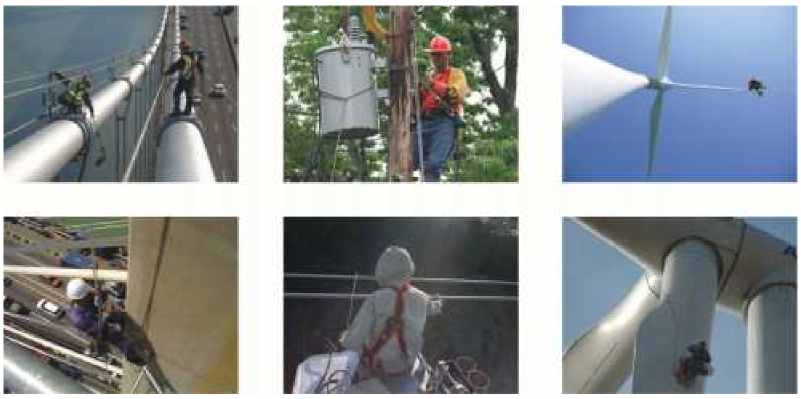


Figure 1.2: Applications scenario in which aerial manipulation might be useful [2]

applications such as infrastructure inspection, valve turning, door opening and drawer operation have been realized.

A third category of aerial manipulation missions consists of assembly and structural construction. In this area different UAMs were used together in teams for building simple truss-like structures leading the way of a new paradigm of aerial workers teams.

A number of other aerial manipulation applications that do not fall into the above categories have also been achieved. For instance, some researches have proposed aerial missions for autonomous water sampling, forest canopy sampling, and sprinkling dispersants over areas affected by oil spills.

In general UAMs are intended to substitute or assist human manpower in all those applications that are considered very dangerous or very costly due to the necessity of highly skilled professionals. Some of these tasks, such as wind turbine or electric line maintenance, are pictured in Figure 1.2.

Scenarios

Two main operational scenario are observed in aerial manipulation.

The more common scenario is a sequential one. The UAM first flies towards an object (to be manipulated) while locking the manipulation mechanism in a configuration that minimizes its disturbance effect on UAV motion. Once the UAM is in close vicinity of the object, the manipulation mechanism is actuated to grasp the object and manipulates it accordingly. For UAMs with simple grippers, the aerial platform flies over the object and once the object is within its reach, the gripper is actuated to grasp the object. For UAMs with manipulators, the UAM tries to maintain its hovering condition while the manipulator is actuated to reach and grasp the object. This phase is the one analysed in this work. Locking the manipulator during the first phase of the mission simplifies motion control problem of the system but has the drawback of extended operation time cost.

In the second scenario, the concurrent one, a UAV and its manipulator operate simultaneously to reduce the required time to carry out a given aerial manipulation task. Some authors experimentally proved that this method can significantly improve the time required to perform a pick-and-place task. The manipulator and/or tether dynamics can be exploited in this approach to improve hovering and tracking performance of a UAV and to significantly extend flight envelope. In conclusion, as compared to the sequential method, concurrent operation of the UAM subsystems presents several advantages and might be adopted more widely in the future with improved dynamic modelling methods and control algorithms.

1.2 Challenges in aerial manipulation

The most important issue to face when dealing with aerial manipulators is related to the movements of the UAV (flying base) caused by the reaction wrenches (force + torque) between it and the operating manipulation mechanism. This means that if one wants to control effectively the position of the manipulator end-effector in order to follow some prescribed trajectory, will have somehow to predict these base movements. As is seen in this thesis, even requiring the end-effector to follow simple linear trajectories causes significant UAV oscillations around the hovering position. In this work the approach consists in predicting the UAV translations and rotations through the momentum conservation law and therefore giving the manipulator motors coherent velocity or acceleration reference inputs. However this is only one way of dealing with the problem. Since UAVs rotations and translations are generally caused by system Center of Gravity (CoG) misalignment with respect to UAV CoG, one possibility could be to command a translating mass attached to the system in order to keep the total

CoG aligned. This mass may be a battery in normal UAMs applications.

Another idea is to adopt a dual arm system. This would not only allow to increase the payload capacity and to extend the effective UAM workspace, but would make also possible the partial cancellation of the reaction wrenches induced by one arm over the base using the other as reaction. This solution would allow also the UAM to perform the manipulation task while perching on something (for example on support structures like poles or cables) with the other arm, letting the system to save energy during the mission.

It goes along that, in order to reduce dynamic wrenches, designers are constrained to use low mass and low inertia arms. In this sense also manipulator actuators have to be light and this causes significant technological limitations. Normal servo actuators employed for building lightweight robotic arms, for example, impede the realization of manipulation tasks involving significant interaction forces. Some prototypes of UAMs employed transmission mechanisms like timing pulleys, or rigid bars for reducing the dynamic coupling effect. In this way, in fact, it is possible to place all the actuators very close to the base of aerial platform allowing not to have additional actuators masses to move along the kinematic chain of the manipulator. Another possible solution to this issue is to adopt a parallel mechanism as in delta manipulators even though this would imply a lower workspace compared to a serial architecture. This work is focused on analysing a serial manipulator attached to the UAV.

Another important issue is the limited energetic autonomy of UAMs. The current time of flight of most UAMs are of the order of minutes or tens of minutes, too short for many practical applications. In order to increase this flight time new configurations of aerial manipulators, new sources of energy and even new control and motion planning method will be needed. This energetic problem is another cause of UAMs limited payload and another reason why it is necessary to employ light arms.

Also regarding the control point of view there are several challenges that designers have to face. For example UAVs usually adopted for aerial manipulation are often underactuated platforms with highly non linear coupled dynamics which significantly complicates their control design. UAV control method adopted in this thesis is discussed in Chapter 2. Another important aspect to consider is that UAVs' propulsion system vary in close vicinity of the ground and/or walls and this may lead to undesired movements of the UAM since the control is sending wrong inputs to the propellers motors.

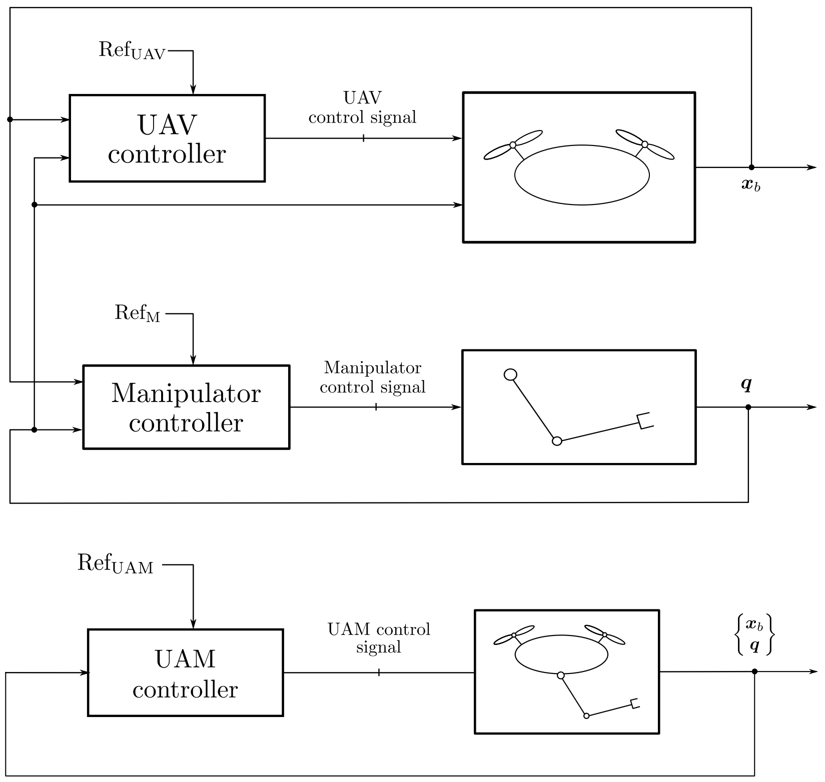
1.3 UAMs control

The primary control objective for UAMs is to drive the attitude and/or position of the UAV and the manipulator end-effector to a desired position/trajectory such that a given mission is accomplished. Position control design of such non linear systems can be categorized as decoupled or coupled, depending on the UAM dynamic model.

In the first category, separate controllers for the UAV and the manipulation mechanism are developed, whereas in the second, the overall system dynamics is considered. For a schematic representation of the two approaches see Fig. 1.3.

1) *Decoupled*: These methods consider the aerial vehicle and the robotic arm as two subsystems that are controlled independently. The decoupled approach relies on the assumption that the influence of the manipulator over the attitude and position dynamics of the aerial platform is relatively small. The dynamic coupling is neglected or at best treated as a disturbance to compensate. This motivates the design of low weight and low inertia manipulators. Decoupled control methods best perform only in quasi-static motions. As soon as the motion is more demanding in terms of accelerations, these methods fail, or in the best case show large tracking errors [3].

2) *Coupled*: These methods consider the system as a unique entity. The design of a coupled control scheme relies on the full dynamic model, which explicitly takes into account the dynamic coupling through the inertia matrix. Therefore, this approach is more suited for dynamic cases and allow better performance in terms of position accuracy and stability. Coupled controllers proposed in the state of the art are strongly model-based and consider the



$\mathrm{Ref}_{\mathrm{UAV}}$
UAV controller
UAV control signal
$\boldsymbol{x}_b$
$\mathrm{Ref}_{\mathrm{M}}$
Manipulator controller
Manipulator control signal
$\boldsymbol{q}$
$\mathrm{Ref}_{\mathrm{UAM}}$
UAM controller
UAM control signal
$\begin{Bmatrix} \boldsymbol{x}_b \\ \boldsymbol{q} \end{Bmatrix}$

Figure 1.3: Schematic diagrams of decoupled and coupled UAM control approaches

full dynamics of the system. A drawback of these control methods is that they require s the real-time computation of the dynamic model of a system with $6+n$ DoFs (Degrees of Freedom), being n the number of manipulator DoFs. Another drawback is that they often require torque controlled motors that are in general unfeasible for aerial manipulators built with conventional servo actuators [3].

3) *Partially coupled*: The control of the aerial platform and the manipulator are independent, but the controllers exploit the information provided by each of the systems to estimate the interaction wrenches and improve the performance of the compound, typically in terms of positioning accuracy. The control strategy presented in this work can fall into this hybrid category.

1.4 Similarities with space manipulation

Space manipulation shares several open issues with aerial manipulation like mobility, teleoperation, and autonomy, additionally coping with extreme environments (microgravity, extreme temperatures, high vacuum, fine dust, high pressure, corrosive atmospheres, radiations, and so on). Hence, many of the methods adopted for free-flying space manipulators can be proposed also in aerial manipulation with the difference that UAMs bases are actuated continuously by the propellers system. The control algorithms that are presented in this work are highly inspired by the pioneering works on space manipulators that were conducted in the late 80s.

Chapter 2

Unmanned Aerial Vehicles

As said, an Unmanned Aerial Vehicles (UAVs) is one of the subsystems composing an aerial manipulator. They are the flying base on which to mount robotic manipulators. In this chapter these devices are discussed in more detail, at first describing their usual fields of application and later presenting their dynamic model and control solutions.

2.1 UAVs applications

The interest in Unmanned Aerial Vehicles has grown a lot in the last decades, not only among researchers and industrial companies, but also among private citizen who could afford these aircrafts. Thanks to their variety and versatility UAVs can be employed in many fields like [5]:

- *Photography and surveillance*: compared to traditional instruments UAVs mobility allow them to have access to normally unreachable environments and to collect footage from unique perspectives. This is the first field of application among amateurs.
- *Safety and rescue*: These devices can reach quickly and without many risks areas hit by catastrophic events for possible inspections that can be useful for rescuing missions.
- *Agricolture*: UAVs are use for farms monitoring and automatic distribution

of fertilizer or automatic sowing.

- *Delivery*: Some companies are experimenting UAVs as aerial shippers for making transportation faster and less impactful on the environment. These nay allow also fast deliveries in case of healthcare emergencies.
- *Archeology*: UAVs can be employed in this field in order to map and make planimetries of archaeological sites.
- *Manipulation*: UAVs equipped with robotic arms or grippers can perform missions in places difficult to reach by human operators. These machines are called aerial manipulators.

2.2 Classification of UAVs

In this section it will presented a brief classification of UAVs based on their morphology. Then it will be explained why multicopters are preferred over the others for aerial manipulation.

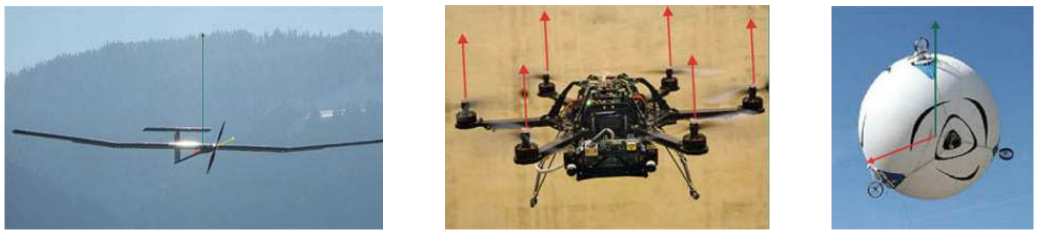
UAVs may vary from different aspects: dimensions, mechanical configuration, actuator types and more. The classification that follows [6] presents UAVs from the most maneuverable with less autonomy to the least maneuverable with longer autonomy:

1) *Rotary-wing UAVs* (RW-UAV): traditional or coaxial helicopters and multicopters (like quadcopter, hexacopter, octocopter...). The latter are tailored to increased maneuverability as well as the ability of stationary vertical flight (hovering).

2) *Convertibles*: like tilt-rotors or cruise-flight-enabled ducted fans. These designs are characterized by the possibility of modifying their configuration for example from the take off to the flight phases.

3) *Bioinspired UAVs*: designs imitating insect. Among these Flapping-wing UAVs are an interesting emerging class.

4) *Fixed-wing UAVs* (FW-UAV): mainly used in agricolture and for mapping



(a) FW-UAV (b) RW-UAV (c) LtA-UAV

Figure 2.1: Different UAV morphologies

5) *Lighter-than-Air UAVs* (LtA-UAV): they use gases lighter than air in order to generate lift.

As anticipated, among these, RW-UAV, and in particular multicopters, are the preferred option for aerial manipulation mainly for three reasons:

- They can keep a stationary position in flight during manipulation missions thanks to their superior maneuverability.
- They are VToL (Vertical Take Off and Landing) aircrafts. Therefore they don't need runways.
- They are relatively inexpensive and are very diffused.

2.3 Multirotor UAVs dynamics

Multirotor UAVs are actuated by two or more motors, usually electric. These vehicles have been very successful because of their simple design capable of being controlled simply by varying the rotational speed of its rotors. This section first presents the characteristics of the drone (an octorotor) that is taken as a reference for the UAM simulations and then describes its mathematical model.

2.3.1 Reference UAV

The drone that has been chosen for being converted to an UAM flying base is the DJI Spreading Wings S1000 (Fig. 2.2) available at University of Padova laboratories. This octorotor, manufactured by the chinese company DJI, is considered very suitable for carrying a manipulator due to its important payload capacity. This model has been designed for high level professional aerial photography and cinematography. In this field it has won the "Technology and Engineering Emmy Awards" in 2017.

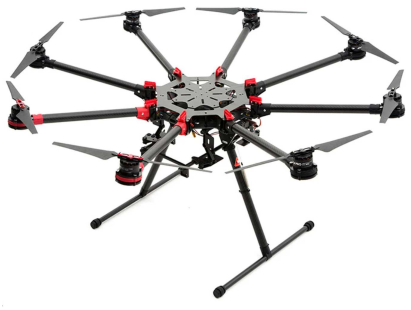


Figure 2.2: DJI Spreading Wings S1000

DJI S1000 consists of an octagonal shape CFRP frame (main body) to which eight arms of 386mm length are attached. At the ends of these arms are located high efficiency 15x5.2 inches propellers, driven by DJI4114 brushless motors. Geometrical data on drone frame can be read in Fig. 2.3.

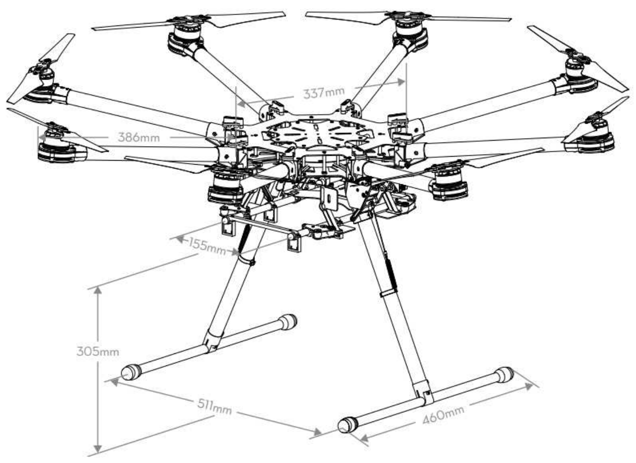


Figure 2.3: DJI S1000 geometry

Drone mass is of approximately 4.2 kg. Moments of inertia were derived through an approximated method in [7] since the manufacturer does not specify them in product manual. These and other inertial parameters are reported in

Table 2.1. For more details see [8].

Total mass (m_{UAV})	4.2 kg
Frame Arm weight (with Landing Gear Mounting Base, Servos)	325 g
Center Frame weight (including Motor, ESC, Propeller)	1330 g
Moment of inertia around horizontal axis (I_{x_B})	0.4097 kg m^2
Moment of inertia around vertical axis (I_{z_B})	0.3421 kg m^2

Table 2.1: DJI S1000 inertial parameters

2.3.2 Mathematical model

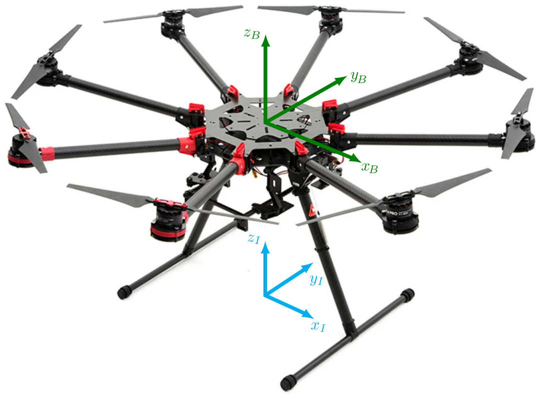


Figure 2.4: DJI S1000: inertial (I) and base (B) reference frames

Octocopters use a set of eight identical propellers with fixed axis of rotation to generate the thrust force for flying. Four of these are thought to rotate clockwise and four counterclockwise. Aircraft control is achieved by varying rotors angular speed and thus modifying lift aerodynamics forces produced by each propeller.

An octorotor is usually catalogued as an under-actuated system, in fact, the control of its 6 Degrees of Freedom during flight is achieved using four control channels. One channel is for the total thrust: through this it is possible to identically increase or decrease the power of all motors to make the UAV to rise or fall

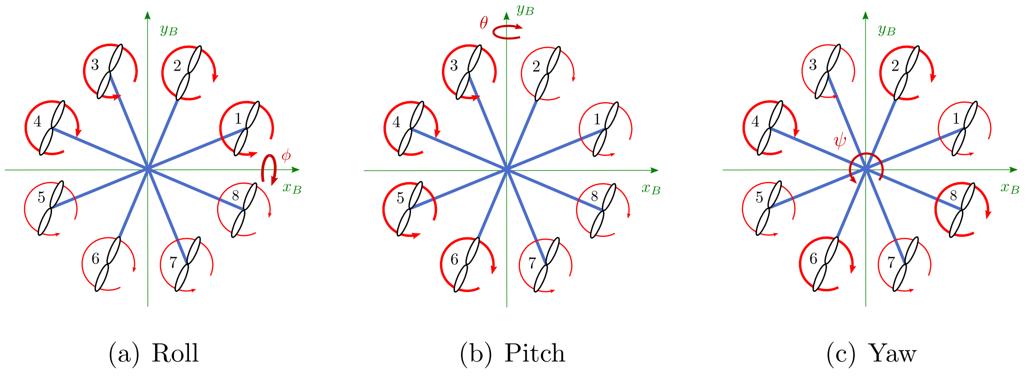


Figure 2.5: Motors inputs for roll, pitch and yaw maneuvers. Thicker lines mean higher velocities

in altitude. The other three channels control roll, pitch and yaw (and so aircraft attitude). In order to perform these last three maneuvers, rotors speed should be adjusted as follows (reference frames as in Fig. 2.4, rotors numbering as in Fig. 2.5):

1) To rotate around x_B axis (*positive roll*) rotational speeds of rotors 1,2,3 and 4 should be increased, decreasing that of the others.

2) To rotate around y_B axis (*positive pitch*) rotational speeds of rotors 3,4,5 and 6 should be increased, decreasing that of the others.

3) To rotate around z_B axis (*positive yaw*) rotational speeds of rotors 2,4,6 and 8 should be increased, decreasing that of the others.

With the assumptions:

- UAV is treated as a rigid body;
- Lift ($\boldsymbol{T}$) and drag ($\boldsymbol{D}$) aerodynamic forces generated by each propeller are considered simply proportional to their angular speeds (Ω);

$$T_i = b\ \Omega_i^2 \qquad D_i = d\ \Omega_i^2$$

- Thrust vectors are considered parallel to z_B axes;

- Drone stays near to hovering conditions;

three-dimensional dynamics of the UAV subjected to weight force is described by:

$$\begin{aligned}
m_{\text{UAV}}\ \ddot{x} &= \phi\ U_1 \\
m_{\text{UAV}}\ \ddot{y} &= \theta\ U_1 \\
m_{\text{UAV}}\ \ddot{y} &= -m_{\text{UAV}} g + U_1 \\
I_x\ \ddot{\theta} &= \dot{\phi}\ \dot{\psi}(I_z - I_x) + I_{z_r}\dot{\phi}\ \Omega - U_2 \\
I_y\ \ddot{\phi} &= \dot{\theta}\ \dot{\psi}(I_y - I_x) + I_{z_r}\dot{\theta}\ \Omega - U_3 \\
I_z\ \ddot{\psi} &= \dot{\phi}\ \dot{\theta}(I_x - I_y) + U_4
\end{aligned} \tag{2.1}$$

with Ω, U_1, U_2, U_3, U_4 defined as:

$$\begin{aligned}
\Omega &= \Omega_1 - \Omega_2 + \Omega_3 - \Omega_4 + \Omega_5 - \Omega_6 + \Omega_7 - \Omega_3 \\
U_1 &= b\,(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2 + \Omega_5^2 - \Omega_6^2 + \Omega_7^2 - \Omega_8^2) \\
U_2 &= b\left[s(\beta_1)\Omega_1^2 + s(\beta_2)\Omega_2^2 + s(\beta_3)\Omega_3^2 + s\left(\beta_4\right)\Omega_4^2 + s\left(\beta_5\right)\Omega_5^2\right. \\
&\quad \left. + s\left(\beta_6\right)\Omega_6^2 + s\left(\beta_7\right)\Omega_7^2 + s\left(\beta_8\right)\Omega_8^2\right] \\
U_3 &= b\left[c(\beta_1)\Omega_1^2 + c(\beta_2)\Omega_2^2 + c(\beta_3)\Omega_3^2 + c\left(\beta_4\right)\Omega_4^2 + c\left(\beta_5\right)\Omega_5^2\right. \\
&\quad \left. c\left(\beta_6\right)\Omega_6^2 + c\left(\beta_7\right)\Omega_7^2 + c\left(\beta_8\right)\Omega_8^2\right] \\
U_4 &= d\ \ (-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2 - \Omega_5^2 + \Omega_5^2 - \Omega_7^2 + \Omega_8^2) - I_{z_r}\Omega
\end{aligned} \tag{2.2}$$

where the symbols mean:

m_{UAV} : total drone mass

I_x, I_y, I_z : diagonal UAV inertia tensor components

I_{z_r} : barycentric propellers moment of inertia around z axis

Ω_i : i-th rotor angular speed magnitude

β_i : angle between x_B axis and and i-th frame arm

b, d : lift and drag propellers coefficients respectively

U_1, U_2, U_3, U_4 are input variables for controlling the UAV. U_1 is the total propellers thrust and its direction depends on UAV inclination which is related to the moments U_2, U_3, U_4 around x_B, y_B, z_B axis respectively. Through these the controller computes rotors angular speeds Ω_i with Eq. 2.3. Ω is not a real input variable even though depends on propellers angular speeds which are input variables. It is related to gyroscopic effects on the UAV and is zero for a drone subjected to weight force in hovering.

$$\begin{Bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{Bmatrix} = \begin{bmatrix} b & b & b & b & b & b & b & b \\ b\,s(\beta_1) & b\,s(\beta_2) & b\,s(\beta_3) & b\,s(\beta_4) & b\,s(\beta_5) & b\,s(\beta_6) & b\,s(\beta_7) & b\,s(\beta_8) \\ b\,c(\beta_1) & b\,c(\beta_2) & b\,c(\beta_3) & b\,c(\beta_4) & b\,c(\beta_5) & b\,c(\beta_6) & b\,c(\beta_7) & b\,c(\beta_8) \\ d & -d & d & -d & d & -d & d & -d \end{bmatrix} \begin{Bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \\ \Omega_5^2 \\ \Omega_6^2 \\ \Omega_7^2 \\ \Omega_8^2 \end{Bmatrix} \tag{2.3}$$

Since the inversion of the matrix in Eq. 2.3 is infeasible, usually controllers compute angular velocities through a mixer law. One possibility is reported in Eq. 2.4 taken from [9]:

$$\Omega_i = U_1 + U_2 \sin(\beta_i) + U_3 \cos(\beta_i) + (-1)^i U_4 \tag{2.4}$$

In conclusion, it is possible to control UAV height and attitude with a controller acting on the variables Ω, U_1, U_2, U_3, U_4. Motion along x and y axes is not independently controllable since the octorotor is under-actuated. Nevertheless it is still possible to follow trajectories in x and y by properly controlling UAV orientation.

2.3.3 Simplified UAV model

In this work the UAV will be modelled as a rigid body controlled by the force U_1 and the moments U_2, U_3, U_4. For the sake of simplicity only planar motion in $x - z$ plane is taken into consideration, with θ unique rotation. In the case of not small angles, UAV equations of motions simplify as:

$$\begin{cases} m_{\mathrm{UAV}}\ \ddot{x} = -s(\theta)\ U_1 \\ m_{\mathrm{UAV}}\ \ddot{y} = -m_{\mathrm{UAV}} g + c(\theta) U_1 \\ I_z\ \ddot{\psi} = U_2 \end{cases} \tag{2.5}$$

2.4 Multicopter control and flight modes

Several control strategies for multicopters have been proposed in literature. As already said, it is not possible to control all 6 UAV DoFs simultaneously due to its under-actuation. Generally this problem is overcome through control systems with a multilayer architecture in which the concentric loops are run at different frequencies [5]. It is possible to distinguish two main loops (see also Fig. 2.6):

1) *Position controller* (external loop): it receive desired aircraft position and yaw as reference inputs and outputs attitude and height references.

2) *Attitude controller* (internal loop): it receives the position controller outputs as inputs and generates the references for the U_i control actions.

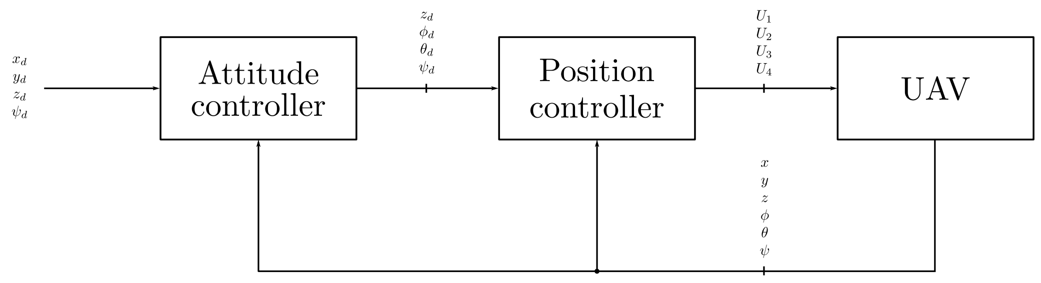

x_d y_d z_d ψ_d
Attitude controller
z_d ϕ_d θ_d ψ_d
Position controller
U_1 U_2 U_3 U_4
UAV
x y z ϕ θ ψ

Figure 2.6: UAV control architecture. Subscript 'd' is for reference quantities

Given a spatial trajectory to be followed and the desired yaw in every instant, the position controller determines the necessary drone tilt angles, thus generating the input for the attitude controller.

Apart from some particular cases where the position controller requires too much computing power, generally control loops are run online by onboard computers which interprets commands coming from the pilot; there may be different flight modes depending on how the commands are handled. The most interesting for aerial manipulation are:

- The control scheme is like the one described above. Pilots commands are UAV position and yaw angle. In this mode if no commands are given by the pilot, the UAV tries to keep its hovering position. If a external disturbance, like wind, acts the platform tilts as to counter the force. In order to adopt this flight mode a position measurement is necessary. Often this is unavailable or unreliable; GPS for example, that is the most used commercial drone positioning system, is unavailable in indoor environments and sometimes provides data with an accuracy of the order of cms; too low for aerial manipulation purposes.
- Position controller is omitted. The pilot commands directly drone height and attitude (gives z_{ref}, ϕ_{ref}, θ_{ref}, ψ_{ref}). This mode does not require an estimation of UAV position so can be used in a wider variety of situations, for example for indoor flights when GPS is unavailable. Its drawback is that, with no commands by the pilot, the UAV is not able to keep its hovering position when external disturbances act on it. Since this mode requires a more skilled pilot it is usually available only in professional multicopters.

Throughout this work only the second flight mode is considered. Considering ($z = 0$, $\theta = 0$) the desired hovering position, both UAV altitude z and rotation θ are controlled by a PID type scheme which will generate the references for the total thrust U_1 and the moment U_2 as:

$$\begin{aligned} U_1 &= k_{P_z} z + k_{D_z} \dot{z} + k_{I_z} \int_0^t z \, dt \\ U_2 &= k_{P_\theta} \theta + k_{D_\theta} \dot{\theta} + k_{I_\theta} \int_0^t \theta \, dt \end{aligned} \tag{2.6}$$

PID control coefficients used for simulations are reported in Table 2.2.

2.4.1 UAVs sensory configuration

The control scheme described above needs an estimation of UAV position, attitude and velocity. In this section follows a brief description of the most important

	k_P	k_I	k_D
Altitude (z)	37 N/m	8 Ns/m	18 N/(m s)
Pitch (θ)	40 Nm/rad	35 Nm s/rad	9 Nm/(rad s)

Table 2.2: UAV PID control coefficients

sensors usually carried by UAVs and that can play an important role also in aerial manipulation.

The most basic instrumentation carried by any multirotor is an Inertial Measurement Unit (IMU) often augmented by some form of height measurements, either acoustic, infrared, barometric, or laser based. Other robotic applications require more sophisticated sensor suites such as VICON systems, Global Positioning System (GPS), cameras, Kinect or scanning laser rangefinder [10]. Due to their limited payload capacity and other reasons (i.e. too expensive instruments) it is impossible for UAVs (and UAMs) to carry the ideal sensory setup needed for every missions. Therefore engineers had to find a way to make the most out of the limited sensors they had. They overcame the limitations of each sensor by relying on the technique known as *sensor fusion*. Thanks to this approach, the control system is provided with informations from the combined sensor array with higher accuracy and refresh rates. The reader can find more on sensor fusion and the Kalman filtering concept in [11].

Inertial Measurement Unit

A typical 6 DOF IMU consists of a three axis accelerometer, a three axis gyroscope and a three axis magnetometer. The first two are inertial sensors and measure the specific force and angular velocity respective to the IMU that can then be related to the UAV body. The axes of both the gyroscope and accelerometer are aligned with the axes of the UAV platform and orthogonal to each other. The magnetometer is used to measure the ambient magnetic field, information that is then used to calculate aircraft heading [11].

The most common type of IMU are Microelectrical Mechanical Systems (MEMS) sensors, due to their low price and small size allowing for a wide variety of use cases and applications.

Sources of error for IMUs include noise and systematic errors such as bias, scaling factors, and misalignment errors. Calibration can alleviate many of the misalignment errors and scaling factor problems. The output forces and angular velocities of these sensors can be estimated as:

$$\begin{aligned} \hat{\boldsymbol{F}}_b &= \boldsymbol{F}_b + \delta \boldsymbol{F}_b + \mathbf{n}_F \\ \hat{\boldsymbol{\omega}}_b &= \boldsymbol{\omega}_b + \delta \boldsymbol{\omega}_b + \mathbf{n}_\omega \end{aligned} \tag{2.7}$$

where $\boldsymbol{F}_b$ and $\boldsymbol{\omega}_b$ are the actual force and angular velocity, respectively. $\mathbf{n}$ is Gaussian white noise (modelled in Section 5.2) and $\delta \boldsymbol{F}_b$ and $\delta \boldsymbol{\omega}_b$ are biases. Update rates for an IMU are on the order of 100–1000 s of Hz. While it is feasible to integrate and obtain position and orientation data, the large influence of error quickly renders these calculations useless. However, through sensor fusion, other sources of information along with the IMU can be combined to generate valid and more accurate pose estimations.

Cameras

Cameras can play a critical role in localization of the aircraft and can become important as well for target acquisition of the object to interact with in aerial manipulation [11].

Among the different techniques used for achieving an accurate pose informations that involve cameras there are Image based Visual Servoing (IBVS), visual odometry and self localization.

For target acquisition, several vision-based detection algorithms (i.e., AR tags and pose detection of circular markers) provide pose measurements of the tracked object in the camera coordinate system, based on its respective projection in the image plane. To estimate both attitude and the position of the UAV, a Bayesian filter can be used to track the target and feed the target's position as a reference for the autopilot. With this approach, it is easy to combine the data from different sensors (i.e., IMU, camera, GPS, motion capture), all coming in at different rates.

GPS

GPS is a satellite-based navigation system with global, persistent, and all-weather coverage. Part of the Global Navigation Satellite System (GNSS) is the most widely used method for location estimation. With a minimum of four satellites within line of sight, a GPS receiver can acquire accurate positioning and timing data anywhere on the earth [11].

In UAVs, GPS is commonly integrated with the on-board IMU and a magnetometer (compass) to determine the vehicle orientation. While GPS does provide accurate information on position, the low sampling rate prevents frequent standalone updates on position required for high-speed applications as well as position derivatives. Further, GPS is susceptible to signal degradation and loss due to interference from buildings, trees, mountains, and other forms of blockage.

Motion Capture systems

Usually, high-level position control system relies on global navigation satellite data fused with on-board visual feedback algorithms. In indoor environments where many scenarios and application take place, a GNSS system probably will not be available. The problem of reliable, robust, and accurate localization can be overcome indoors using a motion capture system. These systems emulate GPS data while relying on visual data to find the target. Motion capture systems are popular pose estimation tools for their high degree of accuracy and high frame rates (systems like VICON can guarantee an accuracy of 50 μm at 375 Hz [10]). They are based on vision markers placed just above the center of mass of the vehicle. An on-board autopilot uses available motion capture data (position and speed) to navigate to a desired set point.

Chapter 3

Robotic manipulators

A manipulator can be schematically represented from a mechanical viewpoint as a kinematic chain of rigid bodies (links) connected by means of revolute or prismatic joints. One end of the chain is usually constrained to a fixed base (fixed-base manipulators), while an end-effector is mounted to the other end (see Fig. 3.1).

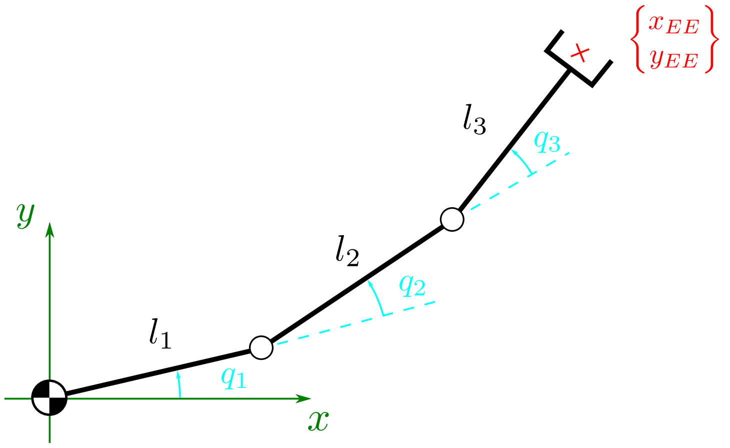


Figure 3.1: 3DoFs fixed-base manipulator

In this chapter the mathematical modeling of both kinematics and dynamics of fixed-base open chain manipulators is presented. This is the foundation for the modeling of free-flying manipulators like UAMs (Chapter 4). Links are always considered rigid. The main reference for this chapter is [12].

3.1 Kinematics of manipulators

This chapter introduces the main notions of manipulator kinematics necessary to describe the relationship between the manipulator configuration and the end-effector (EE) position/orientation (pose). Configuration is described by the n generalized coordinates q_i (n is the number of manipulator DoFs). In this work q_i will always be joint angles since only manipulator with revolute joints will be considered. Joint angles are collected in vector $\boldsymbol{q} \in \mathbb{R}^n$.

There are two problems related to manipulators kinematics: direct and inverse kinematics.

3.1.1 Direct kinematics

The aim of *direct kinematics* is to compute the pose of the end-effector with respect to the base frame as a function of the n joint variables q_i. For an open kinematic chain (only one sequence of links connecting the two ends of the chain) this problem has always a solution that can be found also recursively.

In order to find a recursive solution it is worth defining a coordinate frame attached to each link, from link 0 to link n. In general, the frames can be arbitrarily chosen but usually is used a convention known as *Denavit-Hartenberg* (D-H) convention (see [12] for details). For the aerial manipulator kinematic model a slightly simpler version of D-H approach will be adopted (Chapter 4). Once the frames have been set, the coordinate transformation describing the position and orientation of frame n with respect to frame 0 is represented by:

$$[\boldsymbol{T}_{n,0}(\boldsymbol{q})] = [\boldsymbol{T}_{1,0}(q_1)][\boldsymbol{T}_{2,1}(q_2)]\dots[\boldsymbol{T}_{n,n-1}(q_n)]$$

where $[\boldsymbol{T}_{i,j}]$ is the coordinate transformation matrix from frame $\mathcal{F}_i$ to frame $\mathcal{F}_j$. The actual coordinate transformation describing the position and orientation of the end-effector frame ($\mathcal{F}_e$) with respect to the base frame ($\mathcal{F}_b$) can so be obtained as:

$$[\boldsymbol{T}_{e,b}(\boldsymbol{q})] = [\boldsymbol{T}_{0,b}][\boldsymbol{T}_{n,0}(\boldsymbol{q})][\boldsymbol{T}_{e,n}] \tag{3.1}$$

where $[\boldsymbol{T}_{e,n}]$ is always a constant transformation while $[\boldsymbol{T}_{0,b}]$ is constant only in the case of fixed-base manipulators.

3.1.2 Inverse kinematics

The *inverse kinematic* problem vice versa, is about determining joint variables, and so manipulator configuration, given one or more components of end-effector frame position and orientation. This problem presents several difficulties differently from the direct problem:

- The equations to solve e are in general nonlinear, and thus it is not always possible to find a closed-form solution.
- Multiple solutions may exist.
- Infinite solutions may exist, e.g., in the case of a kinematically redundant manipulator.
- There might be no admissible solutions, in view of the manipulator kinematic structure.

Closed-form solutions of this problem are not always available. In those cases it might be appropriate to resort to numerical solution techniques.

In the following Section, it will be shown how suitable algorithms utilizing the manipulator Jacobian can be employed to solve the inverse kinematics problem.

3.2 Differential kinematics of manipulators

Differential kinematics gives the relationship between joint velocities and the corresponding end-effector linear and angular velocity. This mapping is described by a matrix, termed *geometric Jacobian*, which depends on the manipulator configuration. Alternatively, if the end-effector pose is expressed with reference to a minimal representation in the operational space, it is possible to compute the Jacobian matrix via differentiation of the direct kinematics function with respect to the joint variables. The resulting Jacobian, termed *analytical Jacobian*, in general differs from the geometric one. For planar systems the two Jacobians are identical, so in this thesis this matrix is called $[\boldsymbol{J}]$ with no further specification.

The Jacobian constitutes one of the most important tools for manipulator characterization; in fact, it is useful for finding singularities, analyzing redundancy, determining inverse kinematics algorithms and in many other cases.

3.2.1 Direct differential kinematics

Consider an n-DoFs manipulator. The direct kinematic equation (3.1) can be written in the form:

$$[\boldsymbol{T}_e(\boldsymbol{q})] = \left[\begin{array}{c|c} [\boldsymbol{R}_e(\boldsymbol{q})] & \boldsymbol{r}_e(\boldsymbol{q}) \\ \hline \mathbf{0}^T & 1 \end{array}\right] \tag{3.2}$$

where $[\boldsymbol{R}_e(\boldsymbol{q})]$ is the rotation matrix between end-effector and base frames while $\boldsymbol{r}_e(\boldsymbol{q})$ is represents end-effector position. The goal of direct differential kinematics is to compute end-effector frame linear ($\boldsymbol{v}_e$) and angular ($\boldsymbol{\omega}_e$) velocities given the joints velocities $\dot{\boldsymbol{q}}$. The relations between $\dot{\boldsymbol{q}}$ and $\boldsymbol{v}_e$ and $\boldsymbol{\omega}_e$ are linear in joints velocities and can be written as:

$$\boldsymbol{v}_e = [\boldsymbol{J}_v(\boldsymbol{q})]\dot{\boldsymbol{q}} \tag{3.3}$$

$$\boldsymbol{\omega}_e = [\boldsymbol{J}_\omega(\boldsymbol{q})]\dot{\boldsymbol{q}} \tag{3.4}$$

In (3.3) $[\boldsymbol{J}_v]$ is the $(3\times n)$ matrix relating the contribution of the joint velocities $\dot{\boldsymbol{q}}$ to the end-effector linear velocity $\mathbf{v}_e$, while in (3.4) $[\boldsymbol{J}_\omega]$ is the $(3\times n)$ matrix relating the contribution of the joint velocities $\dot{\boldsymbol{q}}$ to the end-effector angular velocity

$\boldsymbol{\omega}_e$. In compact form, (3.3) and (3.4) can be written as:

$$\mathbf{v}_e = \begin{Bmatrix} \boldsymbol{v}_e \\ \boldsymbol{\omega}_e \end{Bmatrix} = [\boldsymbol{J}(\boldsymbol{q})]\dot{\boldsymbol{q}} \tag{3.5}$$

which represents the *differential kinematic equation.* Note that the Jacobian matrix is a function of joint variables q_i.

The time differentiation of (3.5) leads to the second-order differential kinematics equation:

$$\boldsymbol{a}_e = [\boldsymbol{J}(\boldsymbol{q})]\ddot{\boldsymbol{q}} + [\dot{\boldsymbol{J}}(\boldsymbol{q}, \dot{\boldsymbol{q}})]\dot{\boldsymbol{q}} \tag{3.6}$$

which relates end-effector linear and angular acceleration to joints velocities and accelerations.

3.2.2 Inverse differential kinematics

As mentioned in Section 3.1.2 the inverse kinematics problem admits closed-form solutions only for manipulators having a simple kinematic structure. Problems arise whenever the end-effector attains a particular position and/or orientation in the operational space, or the structure is complex and it is not possible to relate the end-effector pose to different sets of joint variables, or else the manipulator is redundant. These limitations are caused by the highly nonlinear relationship between joint space variables and operational space variables.

On the other hand, the differential kinematics equation represents a linear mapping between the joint velocity space and the operational velocity space, although it varies with the current configuration. This fact suggests the possibility to utilize the differential kinematics equation to tackle the inverse kinematics problem. Suppose that a motion trajectory is assigned to the end-effector in terms of $\mathbf{v}_e$ and the initial conditions on position and orientation. The aim is to determine a feasible joint trajectory $(\boldsymbol{q}, \dot{\boldsymbol{q}})$ that reproduces the given trajectory.

If the manipulator is not redundant and in a non-singular configuration, the joint velocities can be obtained via simple inversion of the Jacobian matrix:

$$\dot{\boldsymbol{q}} = [\boldsymbol{J}(\boldsymbol{q})]^{-1}\mathbf{v}_e \tag{3.7}$$

If the initial manipulator posture $\boldsymbol{q}(0)$ is known, joint positions can be computed by integrating velocities over time i.e.,

$$\boldsymbol{q}(t) = \int_0^t \dot{\boldsymbol{q}}(\zeta) d\zeta + \boldsymbol{q}(0) \tag{3.8}$$

The integration can be performed in discrete time by resorting to numerical techniques. One of the simplest is Euler integration method; given an integration interval Δt, if the positions at time t_k are known the joint positions at time $t_{k+1} = t_k + \Delta t$ can be computed as:

$$\boldsymbol{q}(t_{k+1}) = \boldsymbol{q}(t_k) + \dot{\boldsymbol{q}}(t_k)\Delta t \tag{3.9}$$

This is the integration technique that is adopted in the simulation environment presented in Chapter 4.

If $[\boldsymbol{J}]$ is square and non-singular, it is also possible to solve for joints accelerations by inverting (3.6) getting:

$$\ddot{\boldsymbol{q}} = [\boldsymbol{J}(\boldsymbol{q})]^{-1}(\boldsymbol{a}_e - [\dot{\boldsymbol{J}}(\boldsymbol{q}, \dot{\boldsymbol{q}})]\dot{\boldsymbol{q}}) \tag{3.10}$$

Note that (3.7) and (3.10) work only if the Jacobian is invertible. This might not be true if the assumptions made are not valid. The case of redundant manipulator is discussed in the next Section.

3.2.3 Kinematic redundancy

Let n be the number of kinematic structure DoFs, m the number of variables necessary to characterize the operational space and r the number of operational space variables necessary to specify the manipulator task (i.e. $r = 2$ in case the task consists in the end-effector frame origin tracking a 2D trajectory). The manipulator is said to be intrinsically redundant if $n > m$ and functionally redundant if it has to perform a task so that $r < n$.

The differential kinematic equation can be written formally as (3.5):

$$\mathbf{v}_e = [\boldsymbol{J}(\boldsymbol{q})]\dot{\boldsymbol{q}} \tag{3.11}$$

but now $\mathbf{v}_e$ is meant to be the $(r \times 1)$ vector of end-effector of end-effector velocity of concern for the specific task and $[\boldsymbol{J}]$ is the corresponding $(r \times n)$ matrix that can be extracted from the geometric Jacobian; $\boldsymbol{q}$ is still the $(n \times 1)$ vector of joint velocities.

Since $[\boldsymbol{J}]$ is not square, its inversion is impossible and the inverse differential kinematics, that now presents an infinite number of solution, cannot be solved through 3.7. A viable solution method is to formulate the problem as a constrained linear optimization problem.

In detail, once the end-effector velocity $\mathbf{v}_e$ and Jacobian $[\boldsymbol{J}]$ are given (for a given configuration q), it is desired to find the solutions $\dot{\boldsymbol{q}}$ that satisfy the linear equation in (3.11) and minimize the quadratic cost functional of joint velocities:

$$g(\dot{\boldsymbol{q}}) = \frac{1}{2}(\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_0)^T[\boldsymbol{W}](\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_0) \tag{3.12}$$

where $[\boldsymbol{W}]$ is a suitable $(n \times n)$ symmetric positive definite weighting matrix. This problem can be solved through *Lagrange multipliers* method. The modified cost functional is:

$$g(\dot{\boldsymbol{q}}, \boldsymbol{\lambda}) = \frac{1}{2}(\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_0)^T[\boldsymbol{W}](\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_0) + \boldsymbol{\lambda}^T(\mathbf{v}_e - [\boldsymbol{J}]\dot{\boldsymbol{q}}) \tag{3.13}$$

where $\boldsymbol{\lambda}$ is an $(r \times 1)$ vector of unknown multipliers. Posing

$$\left(\frac{\partial g}{\partial \dot{\boldsymbol{q}}}\right)^T = \mathbf{0} \qquad \left(\frac{\partial g}{\partial \boldsymbol{\lambda}}\right)^T = \mathbf{0}$$

it is possible to solve for joint velocities as:

$$\begin{aligned}\dot{\boldsymbol{q}} = {} & [\boldsymbol{W}]^{-1}[\boldsymbol{J}]^T([\boldsymbol{J}][\boldsymbol{W}]^{-1}[\boldsymbol{J}]^T)^{-1}\mathbf{v}_e + \\ & + ([\mathbb{I}][\boldsymbol{W}]^{-1}[\boldsymbol{J}]^T([\boldsymbol{J}][\boldsymbol{W}]^{-1}[\boldsymbol{J}]^T)^{-1}[\boldsymbol{J}])(\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_0)^T[\boldsymbol{W}](\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_0)\end{aligned} \tag{3.14}$$

where $\mathbb{I}$ represents the identity matrix. The first term of (3.14) represents the solution that minimizes $\dot{\boldsymbol{q}}^T[\boldsymbol{W}]\dot{\boldsymbol{q}}$. In the case of $[\boldsymbol{W}]$ being an identity matrix, the minimum norm joint velocities vector $\dot{\boldsymbol{q}}$. The second term represents a velocity vector that do not alter end-effector velocity and that can be employed for optimizing other objectives choosing $\dot{\boldsymbol{q}}_0$ properly.

If $[\boldsymbol{W}]$ is the identity matrix the solution that minimizes $\dot{\boldsymbol{q}}$ norm is:

$$\dot{\boldsymbol{q}} = [\boldsymbol{J}]^T([\boldsymbol{J}][\boldsymbol{J}]^T)^{-1}\mathbf{v}_e = [\boldsymbol{J}]^\dagger \mathbf{v}_e \tag{3.15}$$

where the matrix $[\boldsymbol{J}]^\dagger = [\boldsymbol{J}]^T([\boldsymbol{J}][\boldsymbol{J}]^T)^{-1}\mathbf{v}_e$ is the *right pseudo-inverse* of $[\boldsymbol{J}]$.

3.3 Dynamics of manipulators

Derivation of the dynamic model of a manipulator plays an important role for simulation of motion, analysis of manipulator structures, and design of control algorithms. Simulating manipulator motion allows control strategies and motion planning techniques to be tested without the need to use a physically available system.

The goal of this section is to present two methods for derivation of the equations of motion of a manipulator in the joint space. The first method is based on the Lagrange formulation and is conceptually simple and systematic. The second method is based on the Newton–Euler formulation and yields the model in a recursive form; it is computationally more efficient since it exploits the typically open structure of the manipulator kinematic chain.

3.3.1 Lagrange formulation

The dynamic model of a manipulator provides a description of the relationship between the joint actuator torques $\boldsymbol{\tau}$ and the motion of the structure. With *Lagrange formulation*, the equations of motion can be derived in a systematic way independently of the reference coordinate frame.

Let $\mathcal{L}$ be the *Lagrangian* of the system, a function of generalized coordinates given by

$$\mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \mathcal{T}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \mathcal{U}(\boldsymbol{q}) \tag{3.16}$$

with $\mathcal{T}$ the *kinetic energy* and $\mathcal{U}$ the *potential energy*. Let $\boldsymbol{\xi}$ be the vector of the *generalized forces* associated to the generalized coordinates $\boldsymbol{q}$. The Lagrange equations can be written in compact form as

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}}\right)^T - \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{q}}\right)^T = \boldsymbol{\xi} \tag{3.17}$$

Equations 3.17 establish the relations existing between the generalized forces applied to the manipulator and the joint positions, velocities and accelerations. Hence, they allow the derivation of the dynamic model of the manipulator starting from the determination of kinetic energy and potential energy of the mechanical system.

After some mathematical steps, that the interested reader can find detailed in [12], equations 3.17 lead to the *joint space dynamic model* of the manipulator that can be written in matrix form as:

$$[\boldsymbol{M}(\boldsymbol{q})]\ddot{\boldsymbol{q}} + [\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})]\dot{\boldsymbol{q}} + [\boldsymbol{F}_v]\dot{\boldsymbol{q}} + [\boldsymbol{F}_s]\mathbf{sgn}(\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} - [\boldsymbol{J}(\boldsymbol{q})]^T \boldsymbol{F}_e \tag{3.18}$$

with:

- $[\boldsymbol{M}(\boldsymbol{q})]$ $(n \times n)$: inertia matrix of the manipulator. It's symmetric and positive definite.
- $[\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})]$ $(n \times n)$: Coriolis matrix. Multiplied by $\dot{\boldsymbol{q}}$ gives the $(n \times 1)$ vector of Coriolis and centrifugal terms.
- $[\boldsymbol{F}_v]$ $(n \times n)$: diagonal matrix of viscous friction coefficients.
- $[\boldsymbol{F}_s]$ $(n \times n)$: diagonal matrix of static friction coefficients (Coulomb model)
- $\boldsymbol{g}(\boldsymbol{q})$ $(n \times 1)$: vector of gravity torques
- $\boldsymbol{\tau}$ $(n \times 1)$: vector of actuating torques
- $[\boldsymbol{J}(\boldsymbol{q})]^T \boldsymbol{F}_e$ $(n \times 1)$: portion of actuating torques spent for balancing external forces on the end-effector
- $\boldsymbol{F}_e = \{\boldsymbol{f}_e, \boldsymbol{m}_e\}^T$ $(n \times 1)$: vector of forces and moments exerted by the end-effector on the environment.

In this work friction forces will be neglected. Equation 3.18 can be expressed in a simplified way as:

$$[\boldsymbol{M}(\boldsymbol{q})]\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{\tau} - [\boldsymbol{J}(\boldsymbol{q})]^T \boldsymbol{F}_e \tag{3.19}$$

where $\boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}})$ comprehends Coriolis, centrifugal and gravity torques.

3.3.2 Newton-Euler formulation

In the Lagrange formulation, the manipulator dynamic model is derived starting from the total Lagrangian of the system. On the other hand, the Newton–Euler formulation is based on a balance of all the forces acting on the generic link of the manipulator. This leads to a set of equations whose structure allows a recursive type of solution; a *forward recursion* is performed for propagating link velocities and accelerations, followed by a *backward recursion* for propagating forces.

Let the symbols be:

- $\boldsymbol{c}_{j,k}$: vector from link j CoG to k-th frame origin
- $[\boldsymbol{I}_{G_i}]_i$: barycentric inertia tensor of link i expressed in frame i
- $\boldsymbol{F}_{j,k}$: Force exerted by link j to link k (mutual action)
- $\boldsymbol{M}_{j,k}$: Moment exerted by link j to link k (mutual force)
- $\boldsymbol{F}_{\text{ext}_i}, \boldsymbol{M}_{\text{ext}_i}$: external forces and moment acting on link i CoG.

Forward recursion

Given $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ and velocity and acceleration of base link $\boldsymbol{a}_{G_0}$, $\boldsymbol{\omega}_0$, $\dot{\boldsymbol{\omega}}_0$, direct kinematics is solved, calculating:

- linear accelerations $\boldsymbol{a}_{G_i}$ of links centroids
- angular velocities $\boldsymbol{\omega}_i$ and accelerations $\dot{\boldsymbol{\omega}}_i$ of links

Forward recursion kinematic formulas are presented in detail in Chapter 4, Section 4.2.1 applied to a multibody system like the UAM.

Backward recursion

Having computed the velocities and accelerations with the forward recursion from the base link to the end-effector, a backward recursion can be carried out for the forces and the moments.

Starting from the last link in the kinematic chain (for which $\boldsymbol{F}_{n,n+1} = \mathbf{0}$), applying the *Newton formula*

$$m_i \boldsymbol{a}_{G_i} = \boldsymbol{F}_{i-1,i} + \boldsymbol{F}_{i,i+1} + m_i \boldsymbol{g}_i + \boldsymbol{F}_{\text{ext}_i} \tag{3.20}$$

to each link, is possible compute the mutual force $\boldsymbol{F}_{i-1,i}$ between the i-th link and the previous. Having computed all the forces it is possible to apply to each link the *Euler formula*

$$\begin{aligned}[\boldsymbol{I}_{G_i}]_i \dot{\boldsymbol{\omega}}_{\boldsymbol{i}} + \boldsymbol{\omega}_{\boldsymbol{i}} \wedge [\boldsymbol{I}_{G_i}]_i \boldsymbol{\omega}_{\boldsymbol{i}} = \boldsymbol{M}_{i-1,i} + \boldsymbol{M}_{i,i+1} + \boldsymbol{M}_{\text{ext}_i} \\ + \boldsymbol{c}_{i,i} \wedge \boldsymbol{F}_{i-1,i} + \boldsymbol{c}_{i,i+1} \wedge \boldsymbol{F}_{i+1,i}\end{aligned} \tag{3.21}$$

in order to compute also the mutual moments $\boldsymbol{M}_{i-1,i}$ backwards. Once the actions exerted by link $i-1$ on link i are known it is possible to compute actuators actions as:

$$\begin{cases} \boldsymbol{\tau}_i(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \boldsymbol{M}_{i-1,i} \cdot \mathbf{z}_i & \text{if } i \text{ is a revolute joint} \\ \boldsymbol{S}_i(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \boldsymbol{F}_{i-1,i} \cdot \mathbf{z}_i & \text{if } i \text{ is a prismatic joint} \end{cases} \tag{3.22}$$

where $\mathbf{z}_i$ represents the unit vector of frame i as in Denavit-Hartenberg convention.

3.3.3 Direct and inverse dynamics

Both Lagrange formulation and Newton–Euler formulation allow the computation of the relationship between the joint torques — and, if present, the end-effector forces — and the motion of the structure. A comparison between the two approaches reveals what follows. The Lagrange formulation has the following advantages:

- It is systematic and of immediate comprehension.
- It provides the equations of motion in a compact analytical form containing the inertia matrix, the matrix in the centrifugal and Coriolis forces, and the vector of gravitational forces. Such a form is advantageous for control design
- It is effective if it wished to include more complex mechanical effects such as flexible link deformation

The Newton–Euler formulation has the following fundamental advantage:

- It is an inherently recursive method that is computationally efficient.

In the study of dynamics it is relevant to find the solution to two kinds of problems:

- *Direct dynamics*: determining, for $t > t_0$ the joint accelerations $\ddot{\boldsymbol{q}}(t)$ (and thus $\dot{\boldsymbol{q}}(t), \boldsymbol{q}(t)$) resulting from the given joint torques $\boldsymbol{\tau}(t)$ (and the possible end-effector forces $\boldsymbol{F}_e(t)$) once the initial position $\boldsymbol{q}(t_0)$ and velocities $\dot{\boldsymbol{q}}(t_0)$ are known (initial state of the system). This is useful especially for manipulator simulation.

- *Inverse dynamics*: determining the joint torques $\boldsymbol{\tau}(t)$ which are needed to generate the motion specified by the joint accelerations $\ddot{\boldsymbol{q}}(t)$, velocities $\dot{\boldsymbol{q}}(t)$, and positions $\boldsymbol{q}(t)$, once the possible end-effector forces $\boldsymbol{F}_e(t)$ are known. This is useful for manipulator trajectory planning and control algorithm implementation.

3.4 Control of manipulators

In 3.2.2 was shown how to invert kinematics by using the differential kinematics equation both at velocity (Eq. 3.7) and acceleration (Eq. 3.10) level. In the case of velocity level kinematics, it was shown also how to integrate numerically the equation. Joints positions calculation is given by Eq. (3.9) that can be detailed in:

$$\boldsymbol{q}(t_{k+1}) = \boldsymbol{q}(t_k) + [\boldsymbol{J}(\boldsymbol{q}(t_k))]^{-1}\boldsymbol{v}_e(t_k)\Delta t. \tag{3.23}$$

As can be seen in (3.23) joint velocities are computed by using the inverse of the Jacobian evaluated with the joint variables at the previous instant of time. It follows that the computed joint velocities $\dot{\boldsymbol{q}}$ do not coincide with those satisfying (3.7) in the continuous time. Therefore, reconstruction of joint variables $\boldsymbol{q}$ is entrusted to a numerical integration which involves drift phenomena of the solution; as a consequence, the end-effector pose corresponding to the computed joint variables differs from the desired one.

Even in the case of negligible numerical errors, joint velocities computed by mean of (3.7) would not allow the manipulator end-effector to track the desired trajectory due to inevitable disturbances that come from the real world, and kinematic model uncertainties.

The same reasoning can be made in the case of inverse kinematics at accelerations level.

These inconveniences highlight the necessity of a feedback loop in addition to the forward loop given by inverse kinematics. The feedback control schemes that will be presented in this Section, and that will be inspirational for the control schemes presented in 6, are based on the *operational space error* between the desired ($\boldsymbol{x}_d$) and the actual ($\boldsymbol{x}_e$) position and orientation of the end-effector:

$$\boldsymbol{e} = \boldsymbol{x}_d - \boldsymbol{x}_e \tag{3.24}$$

This error can be computed by mean of direct kinematics or thanks to visuals sensors like cameras or motion capture systems that track end-effector position. In the case of second-order algorithms also the time derivative of the error,

$$\dot{\boldsymbol{e}} = \dot{\boldsymbol{x}}_d - \dot{\boldsymbol{x}}_e \tag{3.25}$$

(i.e. the end-effector velocity error) plays an important role.

In these types of schemes the outputs are kinematic quantities like motors velocities or accelerations. For this reason they are called CLIK algorithms, i.e. Closed-Loop Inverse Kinematic algorithms. In the following sections both first-order and second-order algorithms are presented.

3.4.1 First-order kinematic algorithms

In these schemes the manipulator is controlled at the velocity level. This means that the algorithms gives as outputs the velocity profiles $\dot{\boldsymbol{q}}$ to be followed by the arm servo-motors.

According to differential kinematics (3.5) the time derivative of operational space error (3.24) can be rewritten as:

$$\dot{\boldsymbol{e}} = \dot{\boldsymbol{x}}_e - [\boldsymbol{J}]\dot{\boldsymbol{q}}. \tag{3.26}$$

In order to get an inverse kinematic algorithm that works, $\dot{\boldsymbol{q}}$ has to be chosen as a function of error so that Eq. (3.26) describes a system converging to zero. Eq. (3.26) is also called dynamic error equation since it describes the evolution of the error over time.

Jacobian (pseudo-) inverse algorithm

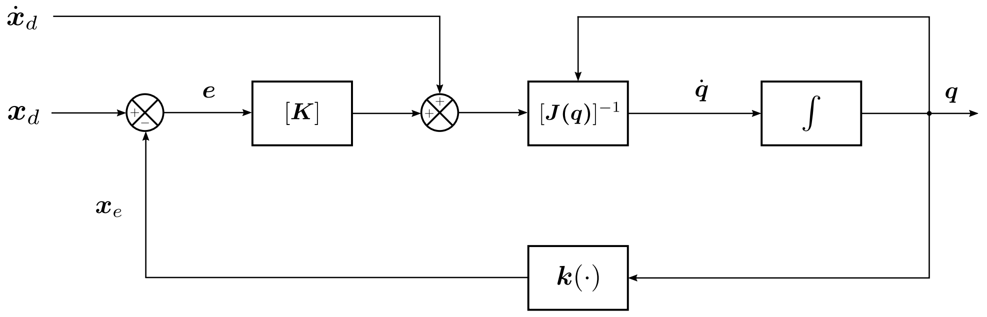


Figure 3.2: Inverse Jacobian algorithm

In this scheme the control velocity is computed as:

$$\dot{\boldsymbol{q}} = [\boldsymbol{J}(\boldsymbol{q})]^{-1} (\dot{\boldsymbol{x}}_d - [\boldsymbol{K}]\boldsymbol{e}) \tag{3.27}$$

where $[\boldsymbol{K}]$ is a (usually diagonal) matrix of gains. With this choice of velocity inputs the dynamic error equation is:

$$\boldsymbol{e} + [\boldsymbol{K}]\dot{\boldsymbol{e}} = \boldsymbol{0}. \tag{3.28}$$

If $[\boldsymbol{K}]$ is a positive definite matrix then the system is asymptotically stable. The error tends to zero along the trajectory with a convergence rate that depends on the eigenvalues of matrix $[\boldsymbol{K}]$; the larger the eigenvalues, the faster the convergence. Since the scheme is practically implemented as a discrete-time system, it is reasonable to predict that an upper bound exists on the eigenvalues; depending on the sampling time, there will be a limit for the maximum eigenvalue of $[\boldsymbol{K}]$ under which asymptotic stability of the error system is guaranteed.

The block diagrams corresponding to the scheme (3.27) is illustrated in Fig. 3.2 where $\boldsymbol{k}(\cdot)$ indicates the direct kinematic function $\boldsymbol{x}_e = \boldsymbol{k}(\boldsymbol{q})$

Equation (3.27) can also be rewritten as:

$$\begin{aligned} \dot{\boldsymbol{q}} &= [\boldsymbol{J}(\boldsymbol{q})]^{-1}\dot{\boldsymbol{x}}_d + [\boldsymbol{J}(\boldsymbol{q})]^{-1}[\boldsymbol{K}]\boldsymbol{e} \\ &= \dot{\boldsymbol{q}}_{\text{ref}} + \Delta\dot{\boldsymbol{q}} \end{aligned} \tag{3.29}$$

in which it can be clearly seen that the first term is a feed-forward velocity coming from inverse differential kinematics, and the second is a feedback term. This formulation will reappear in Chapter 6.

In the case of a *redundant manipulator* solution 3.27 can be generalized as:

$$\dot{\boldsymbol{q}} = [\boldsymbol{J}]^{\dagger}(\dot{\boldsymbol{x}}_e + [\boldsymbol{K}]\boldsymbol{e}) + ([\mathbb{I}_n] - [\boldsymbol{J}]^{\dagger}\boldsymbol{J})\dot{\boldsymbol{q}}_0 \tag{3.30}$$

Jacobian transpose algorithm

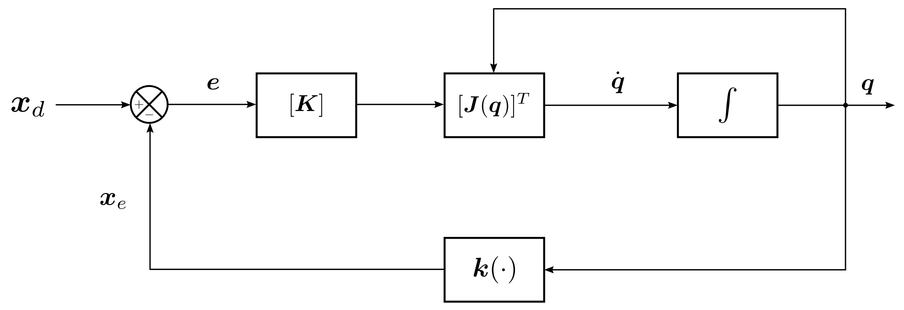


Figure 3.3: Jacobian transpose algorithm

A computationally simpler algorithm can be derived by finding a relationship between $\dot{\boldsymbol{q}}$ and $\boldsymbol{e}$ that ensures error convergence to zero, without requiring linearization of (3.26). In the case of a constant reference ($\dot{\boldsymbol{x}}_d = \boldsymbol{0}$) this scheme proposes the velocity vector

$$\dot{\boldsymbol{q}} = [\boldsymbol{J}(\boldsymbol{q})]^T[\boldsymbol{K}]\boldsymbol{e}. \tag{3.31}$$

The great advantage of this scheme is that it requires only the transposition of the Jacobian, that is a rather simpler operation than the inversion. In the case of not constant reference ($\boldsymbol{x}_d \neq \boldsymbol{0}$) the system can achieve good performances if gains are high enough and the trajectory has not to be tracked to fast. Further discussion on asymptotic stability of this scheme can be found in [12]

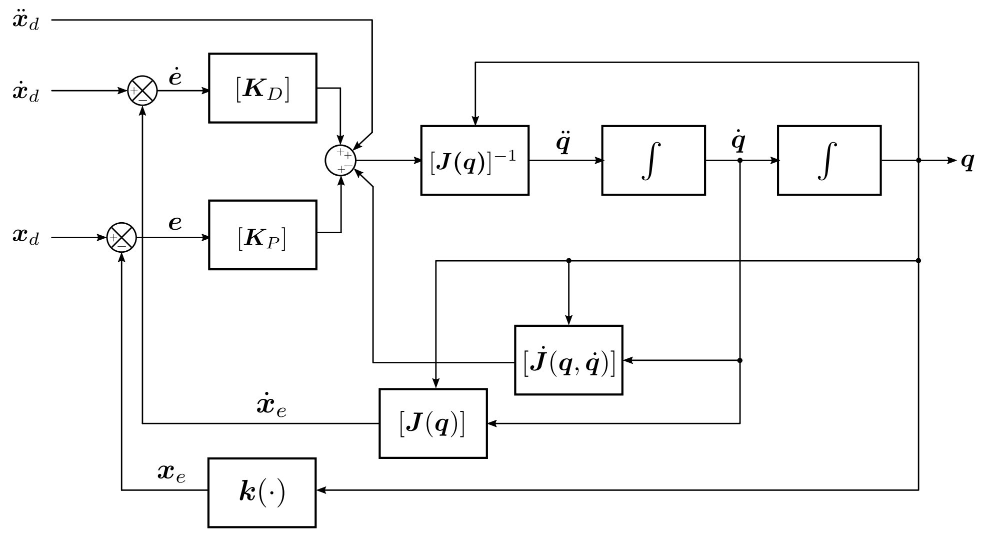


Figure 3.4: Second-order algorithm

3.4.2 Second-order kinematic algorithm

For control purposes it may be necessary to invert a motion trajectory specified in terms of position, velocity and acceleration. As seen in Section 3.2.2, under the assumptions of a square and non singular matrix $[\boldsymbol{J}]$, the second-order differential kinematics can be inverted in terms of joint accelerations as:

$$\ddot{\boldsymbol{q}} = [\boldsymbol{J}(\boldsymbol{q})]^{-1}(\ddot{\boldsymbol{x}}_e - [\dot{\boldsymbol{J}}(\boldsymbol{q},\dot{\boldsymbol{q}})]\dot{\boldsymbol{q}}) \tag{3.32}$$

where in this case the end-effector acceleration has been indicated with $\ddot{\boldsymbol{x}}_e$.

The error defined as:

$$\ddot{\boldsymbol{e}} = \ddot{\boldsymbol{x}}_d - \ddot{\boldsymbol{x}}_e \tag{3.33}$$

leads to the error equation

$$\ddot{\boldsymbol{e}} = \ddot{\boldsymbol{x}}_d - [\boldsymbol{J}(\boldsymbol{q})]\ddot{\boldsymbol{q}} - [\dot{\boldsymbol{J}}(\boldsymbol{q},\dot{\boldsymbol{q}})]\dot{\boldsymbol{q}} \tag{3.34}$$

This control scheme imposes the joint accelerations as:

$$\ddot{\boldsymbol{q}} = [\boldsymbol{J}(\boldsymbol{q})]^{-1}(\ddot{\boldsymbol{x}}_d - [\dot{\boldsymbol{J}}(\boldsymbol{q},\dot{\boldsymbol{q}})]\dot{\boldsymbol{q}} + [\boldsymbol{K}_D]\dot{\boldsymbol{e}} + [\boldsymbol{K}_P]\boldsymbol{e}) \tag{3.35}$$

where $[\boldsymbol{K}_D]$ and $[\boldsymbol{K}_P]$ are positive definite (usually diagonal) matrices. Like for (3.29) the input (3.35) can be decomposed as

$$\ddot{\boldsymbol{q}} = \ddot{\boldsymbol{q}}_{\text{ref}} + \Delta\ddot{\boldsymbol{q}}. \tag{3.36}$$

Substituting (3.35) into (3.34) leads to the equivalent system:

$$\ddot{\boldsymbol{e}} + [\boldsymbol{K}_D]\dot{\boldsymbol{e}} + [\boldsymbol{K}_P]\boldsymbol{e} = \boldsymbol{0} \tag{3.37}$$

which is asymptotically stable.

In the case of a redundant manipulator, the generalization of (3.35) leads to an algorithmic solution based on the Jacobian pseudo-inverse of the kind

$$\ddot{\boldsymbol{q}} = [\boldsymbol{J}(\boldsymbol{q})]^{\dagger}(\ddot{\boldsymbol{x}}_d - [\dot{\boldsymbol{J}}(\boldsymbol{q}, \dot{\boldsymbol{q}})]\dot{\boldsymbol{q}} + [\boldsymbol{K}_D]\dot{\boldsymbol{e}} + [\boldsymbol{K}_P]\boldsymbol{e}) + ([\mathbb{I}_n] + [\boldsymbol{J}(\boldsymbol{q})]^{\dagger}[\boldsymbol{J}(\boldsymbol{q})])\ddot{\boldsymbol{q}}_0 \tag{3.38}$$

Chapter 4

UAMs modeling

4.1 Introduction

A UAM (Unmanned Aerial Manipulator) is a multibody system consisting of a main body with one or more manipulator/s arms attached to. In this Chapter the theory behind the kinematic and dynamic modeling of these kind of systems is presented. The equations underlying these models form the basis of a simulation environment that is used to test the control schemes presented in Chapter 6. This is further enriched by the disturbance models presented in Chapter 5. This environment was entirely written in MATLAB language starting from a library of functions for kinematics and dynamics of free-base robots previously developed in [13]. In Section 4.4 this simulator is validated in ADAMS.

Even though in this work only planar single arm UAMs will be analysed, the theory here presented refers to a more general case of a free-flying robot with multiple serial arms. All the joints connecting links will be considered to be rotational since this is the most common scenario among UAMs.

4.1.1 Mathematical graph representation

In order to mathematically describe the interconnection of the bodies, a method is adopted from mathematical graph theory. The rules on the assignment of links and joints indices are as follows:

- Reference body (UAV body) is denoted as link 0. This will also be referred

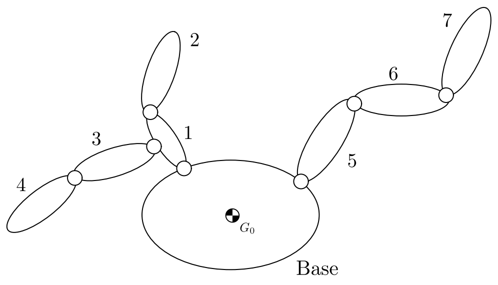


Figure 4.1: Sample system

to with the subscript "b".

- The index of a link i in the phvsical connection between link 0 and link j must be $0 < i < j$.
- One link can have multiple connections with other links. As a result of the above statement, a link i has one *lower connection* (which index is smaller than i) and zero, one or multiple *upper connection(s)* (which index is greater than i).
- There is one single joint interconnecting two links.
- Indices of joints begin from 1, and joint i is physically attached on link i.

After indices assignments a connection index vector $\boldsymbol{B}$ and incidence matrices $[\boldsymbol{S}]$,$[\boldsymbol{S}_0]$,$[\boldsymbol{S}_e]$ are constructed.

The connection index vector $\boldsymbol{B}$ is used to find the lower connection of a link $i > 0$. The element B_i is the index of the the lower connection of link i.

Incidence matrix $[\boldsymbol{S}]$ is used to find the upper connection of a link, which may not exist or exist one or more. Each element S_{ij} is defined as:

$$S_{ij} = \begin{cases} +1 & \text{if } i = B_j \\ -1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

$\boldsymbol{S}_0$ and $\boldsymbol{S}_e$ elements are defined as:

$$S_{0_i} = \begin{cases} +1 & \text{if } 0 = B_j \\ 0 & \text{otherwise} \end{cases}$$

$$S_{e_i} = \begin{cases} +1 & \text{if } i \text{ is a terminal link} \\ 0 & \text{otherwise} \end{cases}$$

Figure 4.1 depicts a general multibody system to which this theory may be applied.

4.1.2 Coordinate systems

Let the stationary inertial coordinate frame be denoted by $\mathcal{F}_I$. On each link are defined moving coordinate frames $\mathcal{F}_i$ adopting a rule slightly different from Denavit-Hartenberg convention. Since only revolute joints are considered the rules are:

- the origin of the coordinate system $\mathcal{F}_i$ is located on joint i and fixed on link i
- $\mathcal{F}_i$ z-axis coincides with joint rotation axis
- x-axis points toward joint $i+1$ or has the direction in which the inertia tensor is expressed easier

Figure 4.2 shows a single arm system with its frames.

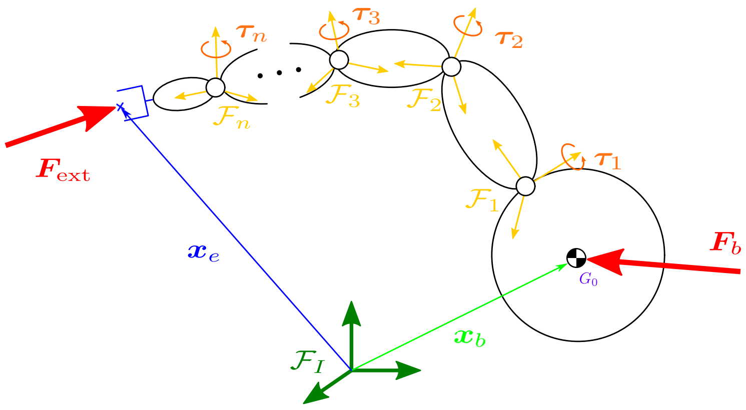


Figure 4.2: Example of single arm free-flying system

4.1.3 Notation used

Symbols descriptions

The symbols that will be used are defined as follows:

- $[\mathbb{I}_n]$: $n \times n$ identity matrix
- $\hat{\mathbf{z}}_i$: unit vector indicating the rotational axis of joint j
- Inertial properties:

  m_{tot}: total mass of the system

  m_i: mass of link i

  $[\boldsymbol{I}_{G_i}]$: inertia tensor of link i with respect to its center of mass G_i
- Positions:

  $\boldsymbol{r}_{G_0}$ $(= \boldsymbol{r}_{G_b})$: base center of mass position vector

  $\boldsymbol{\phi}_b = \{\phi_0, \theta_0, \psi_0\}^T$: vector of roll, pitch and yaw angles of base body

  $\boldsymbol{r}_{G_i}$: link i center of mass position vector

  $\boldsymbol{r}_G$: total system center of mass position vector

  $\boldsymbol{r}_{j_i}$: position vector of joint i

  $\boldsymbol{r}_e$: end-effector position vector

  $\boldsymbol{q}$: vector of joint angles
- Velocities and accelerations:

  $\boldsymbol{v}_{G_0}$, $\dot{\boldsymbol{v}}_{G_0}$ $(= \boldsymbol{v}_b$, $\dot{\boldsymbol{v}}_b)$: base body center of mass velocity and acceleration

  $\boldsymbol{v}_{G_i}$, $\dot{\boldsymbol{v}}_{G_i}$: link i center of mass velocity and acceleration

  $\boldsymbol{\omega}_i$, $\dot{\boldsymbol{\omega}}_i$: link i angular velocity and acceleration

  $\mathrm{v}_b = \{\boldsymbol{v}_b, \boldsymbol{\omega}_b\}^T = \{\boldsymbol{v}_{G_0}, \boldsymbol{\omega}_0\}$: base twist

  $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$: vectors of joint velocities and accelerations
- Forces/moments on the system:

$\boldsymbol{F}_b = \{\boldsymbol{f}_b, \boldsymbol{m}_b\}^T$: external forces and moments on the base

$\boldsymbol{F}_e = \{\boldsymbol{f}_e, \boldsymbol{m}_e\}^T$: external forces and moments on manipulator hand

$\boldsymbol{\tau}$: joint torques vector

For each link are then defined some "link vectors" (see Figure 4.3) as:

- $\boldsymbol{c}_{i,j}$: vector from the center of mass of link i to link j
- $\boldsymbol{\ell}_{i,j}$: vector from joint i to joint j

$$\boldsymbol{\ell}_{i,j} = \boldsymbol{c}_{i,j} - \boldsymbol{c}_{i,i} \tag{4.1}$$

- $\boldsymbol{c}_{ie}$: vector from link i center of mass to the end-point if link i is an end-link
- $\boldsymbol{\ell}_{ie}$: vector from joint i to the end-point

$$\boldsymbol{\ell}_{ie} = \boldsymbol{c}_{ie} - \boldsymbol{c}_{i,i} \tag{4.2}$$

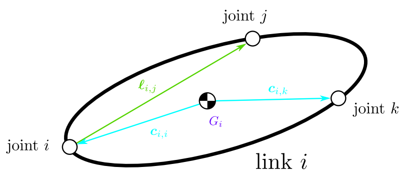


Figure 4.3: Link vectors

Skew operator

The operator $[\boldsymbol{S}(\cdot)]$ for a vector $\boldsymbol{a} = \{a_x, a_y, a_z\}^T$ is called *skew* operator and the matrix

$$[\boldsymbol{S}(\boldsymbol{a})] = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{4.3}$$

is called skew-symmetric matrix. It has the property

$$\boldsymbol{a} \wedge \boldsymbol{b} = [\boldsymbol{S}(\boldsymbol{a})]\boldsymbol{b}. \tag{4.4}$$

It will be used to express in matrix form the vectorial product.

Notation for projected vectorial equations

In the following sections vectorial equations may be projected onto some particular reference frame (usually the inertial reference $\mathcal{F}_I$). $\{a\}_j$ will indicate the projection of the generic vector $\boldsymbol{a}$ onto frame j and $[R_{jk}]$ the rotation matrix for converting the coordinates from reference j to reference k.

4.2 Kinematics

4.2.1 Link velocities and accelerations

For successive links connected by a revolute joint, velocities $\boldsymbol{v}_{G_i}$ and $\boldsymbol{\omega}_i$ can be calculated recursively, as anticipated in 3.3.2, as:

$$\{\omega_i\}_I = \{\omega_{B_i}\}_I + [R_{iI}]\{\hat{z}_i\}_i\dot{q}_i \tag{4.5}$$

$$\{v_{G_i}\}_I = \{v_{B_i}\}_I + \{\omega_{B_i}\}_I \wedge \{c_{B_i,i}\}_I - \{\omega_i\}_I \wedge \{c_{i,i}\}_I \tag{4.6}$$

and accelerations $\dot{\boldsymbol{v}}_{G_i}$, $\dot{\boldsymbol{\omega}}_i$ as:

$$\{\dot{\omega}_i\}_I = \{\dot{\omega}_{B_i}\}_I + \{\omega_i\}_I \wedge ([R_{iI}]\{z_i\}_i\dot{q}_i) + [R_{iI}]\{\hat{z}_i\}_i\ddot{q}_i \tag{4.7}$$

$$\begin{aligned}\{\dot{v}_{G_i}\}_I = \{\dot{v}_{B_i}\}_I + \{\dot{\omega}_{B_i}\}_I \wedge \{c_{B_i,i}\}_I + \{\omega_{B_i}\}_I \wedge (\{\omega_{B_i,i}\}_I \wedge \{c_{B_i,i}\}_I)\\ - \{\dot{\omega}_i\}_I \wedge \{c_{i,i}\}_I - \{\omega_i\}_I \wedge (\{\omega_i\}_I \wedge \{c_{i,i}\}_I)\end{aligned} \tag{4.8}$$

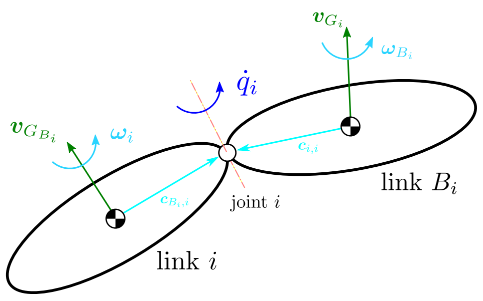


Figure 4.4: Kinematics of the multibody system

The velocities of the i-th link ($i > 0$) can also be represented in the forms:

$$\boldsymbol{v}_{G_i} = [\boldsymbol{J}_v]_i \dot{\boldsymbol{q}} \tag{4.9}$$

$$\boldsymbol{\omega}_i = [\boldsymbol{J}_\omega]_i \dot{\boldsymbol{q}} \tag{4.10}$$

where $[\boldsymbol{J}_v]_i$, $[\boldsymbol{J}_\omega]_i \in \mathbb{R}^{3\times n}$ are the Jacobian matrices of i-th body given by

$$[\boldsymbol{J}_v]_i = [\hat{\mathbf{z}}_1 \wedge (\boldsymbol{r}_{G_i} - \boldsymbol{r}_{j_1}), \hat{\mathbf{z}}_2 \wedge (\boldsymbol{r}_{G_i} - \boldsymbol{r}_{j_2}), \cdots, \hat{\mathbf{z}}_i \wedge (\boldsymbol{r}_{G_i} - \boldsymbol{r}_{j_i}), \mathbf{0}, \cdots, \mathbf{0}] \tag{4.11}$$

$$[\boldsymbol{J}_\omega]_i = [\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2, \cdots, \hat{\mathbf{z}}_i, \mathbf{0}, \cdots, \mathbf{0}] \tag{4.12}$$

4.2.2 End-points kinematics

The kinematic relationship relating the end-points velocities/accelerations to the generalized coordinates are:

$$\dot{\boldsymbol{x}}_e = [\boldsymbol{J}_b]\dot{\boldsymbol{x}}_b + [\boldsymbol{J}_m]\dot{\boldsymbol{q}} \tag{4.13}$$

$$\ddot{\boldsymbol{x}}_e = [\boldsymbol{J}_b]\ddot{\boldsymbol{x}}_b + [\dot{\boldsymbol{J}}_b]\dot{\boldsymbol{x}}_b + [\boldsymbol{J}_m]\ddot{\boldsymbol{q}} + [\dot{\boldsymbol{J}}_m]\dot{\boldsymbol{q}} \tag{4.14}$$

with

- $\boldsymbol{x}_e \in \mathbb{R}^6$: position/orientation of end-point
- $\boldsymbol{x}_b \in \mathbb{R}^6$: position/orientation of base body ($\dot{\boldsymbol{x}}_b = \{\boldsymbol{v}_b, \dot{\boldsymbol{\phi}}_b\}^T$)
- $[\boldsymbol{J}_b] \in \mathbb{R}^{6\times 6}$: Jacobian matrix for base variables. Its expression is

$$[\boldsymbol{J}_b] = \begin{bmatrix} [\mathbb{I}_3] & -[\boldsymbol{S}(\boldsymbol{r}_{G_0,e})] \\ [\mathbf{0}] & [\mathbb{I}_3] \end{bmatrix} \quad , \quad \boldsymbol{r}_{G_0,e} = \boldsymbol{r}_e - \boldsymbol{r}_{G_0} \tag{4.15}$$

- $[\boldsymbol{J}_m] \in \mathbb{R}^{6\times n}$: Jacobian matrix for joints variables. This is just a Jacobian for ground-based manipulators, however, described from the inertial frame. Its expression is

$$[\boldsymbol{J}_m] = \begin{bmatrix} \hat{\mathbf{z}}_1 \wedge (\boldsymbol{r_e} - \boldsymbol{r}_{j_1}) & \hat{\mathbf{z}}_2 \wedge (\boldsymbol{r_e} - \boldsymbol{r}_{j_2}) & \cdots & \hat{\mathbf{z}}_n \wedge (\boldsymbol{r_e} - \boldsymbol{r}_{j_n}) \\ \hat{\mathbf{z}}_1 & \hat{\mathbf{z}}_2 & \cdots & \hat{\mathbf{z}}_n \end{bmatrix} \tag{4.16}$$

4.3 Dynamics

A UAM can be seen as a spacecraft-manipulator system whose base is controlled. Such a system is also called *free-flying* robot. Thus, UAMs dynamics will be developed using the method conceived for such systems. The reader can find more in [14].

4.3.1 Equations of motion

The equation of motion for a free-flying system can be derived by applying the Lagrangian approach (see 3.3.1). In matrix form they are:

$$\begin{bmatrix} [\boldsymbol{H}_b] & [\boldsymbol{H}_{bm}] \\ [\boldsymbol{H}_{bm}]^T & [\boldsymbol{H}_m] \end{bmatrix} \begin{Bmatrix} \ddot{\boldsymbol{x}} \\ \ddot{\boldsymbol{q}} \end{Bmatrix} + \begin{Bmatrix} \boldsymbol{c}_b \\ \boldsymbol{c}_m \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{F}_b \\ \boldsymbol{\tau} \end{Bmatrix} + \begin{bmatrix} [\boldsymbol{J}_b]^T \\ [\boldsymbol{J}_m]^T \end{bmatrix} \boldsymbol{F}_e \tag{4.17}$$

where:

- $[\boldsymbol{H}_b]$ is the *base inertia matrix*
- $[\boldsymbol{H}_{bm}]$ is the base-manipulator *coupling matrix*
- $[\boldsymbol{H}_m]$ is the *manipulator mass matrix* (identical to the one in Eq. 3.18)
- $\boldsymbol{c}_b, \boldsymbol{c}_m$ are velocity dependent non-linear terms (*Coriolis and centrifugal* terms)

The matrices expressions are:

- $$[\boldsymbol{H}_b] \in \mathbb{R}^{6\times 6} = \begin{bmatrix} m_{\text{tot}}[\mathbb{I}_3] & m_{\text{tot}}[\boldsymbol{S}(\boldsymbol{r}_{G_0,G})]^T \\ m_{\text{tot}}[\boldsymbol{S}(\boldsymbol{r}_{G_0,G})] & [\boldsymbol{H}_\omega] \end{bmatrix} \tag{4.18}$$
- $$[\boldsymbol{H}_\omega] \in \mathbb{R}^{3\times 3} = [\boldsymbol{I}_{G_0}] + \sum_{i=1}^{n}([\boldsymbol{I}_{G_i}] + m_i[\boldsymbol{S}(\boldsymbol{r}_{G_0,G_i})]^T[\boldsymbol{S}(\boldsymbol{r}_{G_0,G_i})]) \ ,$$
  $$(\boldsymbol{r}_{G_0,G_i} = \boldsymbol{r}_{G_i} - \boldsymbol{r}_{G_0}) \tag{4.19}$$
- $$[\boldsymbol{H}_{bm}] \in \mathbb{R}^{6\times n} = \begin{bmatrix} [\boldsymbol{J}_v]_{\text{UAM}} \\ [\boldsymbol{H}_{\omega q}] \end{bmatrix} \tag{4.20}$$

- $[\boldsymbol{J}_v]_{\text{UAM}} \in \mathbb{R}^{3\times n} = \sum_{i=1}^{n} m_i [\boldsymbol{J}_v]_i / m_{\text{tot}}$ (4.21)

- $[\boldsymbol{H}_{\omega\phi}] \in \mathbb{R}^{3\times n} = \sum_{i=1}^{n} ([\boldsymbol{I}_{G_i}][\boldsymbol{J}_\omega]_i + m_i[\boldsymbol{S}(\boldsymbol{r}_{G_0,G_i})][\boldsymbol{J}_v]_i)$ (4.22)

- $[\boldsymbol{H}_m] \in \mathbb{R}^{n\times n} = \sum_{i=1}^{n} ([\boldsymbol{J}_\omega]_i^T [\boldsymbol{I}_{G_i}][\boldsymbol{J}_\omega]_i + m_i [\boldsymbol{J}_v]_i^T [\boldsymbol{J}_v]_i)$ (4.23)

4.3.2 Linear and angular momenta

Let $\boldsymbol{p}$ be the linear momentum of the whole system and $\boldsymbol{K}_{G_0} \equiv \boldsymbol{K}_{G_{\text{UAV}}} \equiv \boldsymbol{K}$ the angular momentum with respect to the base center of mass. They are collected in the vector $\boldsymbol{h}$. The relationship between $\boldsymbol{h}$ and the generalized coordinates velocities $\{\dot{\boldsymbol{x}}_b, \dot{\boldsymbol{q}}\}^T$ is:

$$\boldsymbol{h} = \begin{Bmatrix} \boldsymbol{p} \\ \boldsymbol{K} \end{Bmatrix} = [\boldsymbol{H}_b]\dot{\boldsymbol{x}}_b + [\boldsymbol{H}_{bm}]\dot{\boldsymbol{q}} \tag{4.24}$$

The time differentiation of 4.24 yields:

$$\frac{d}{dt}\begin{Bmatrix} \boldsymbol{p} \\ \boldsymbol{K} \end{Bmatrix} = \dot{\boldsymbol{h}} = \boldsymbol{F}_{\text{e}} = [\boldsymbol{H}_b]\ddot{\boldsymbol{x}}_b + [\boldsymbol{H}_{bm}]\ddot{\boldsymbol{q}} + [\dot{\boldsymbol{H}}_b]\dot{\boldsymbol{x}}_b + [\dot{\boldsymbol{H}}_{bm}]\dot{\boldsymbol{q}} \tag{4.25}$$

The first equivalence in (4.25) is given by Newton and Euler formulas which say that the time derivative of linear and angular momenta are respectively equal to the external force and moment acting on the system.

4.3.3 Inverse dynamics

Inverse dynamics computation is critical for UAM simulation as will be seen in 4.3.4. It's used for computing the torques $\boldsymbol{\tau}$ that the motors have to provide for tracking the control kinematic references, and for the computation of the non-linear term $\boldsymbol{c}_m$, $\boldsymbol{c}_b$ in Equation 4.17. It can be used also for a computed torque control. For the computation, the classical order-n recursive Newton-Euler approach (see 3.3.2) is adopted. Newton and Euler equations for a link i are expressed as:

$$\boldsymbol{f}_{\text{in},i} = m_i \dot{\boldsymbol{v}}_{G_i} \tag{4.26}$$

$$\boldsymbol{m}_{\text{in},i} = [\boldsymbol{I}_{G_i}]\dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \wedge ([\boldsymbol{I}_{G_i}]\boldsymbol{\omega}_i) \tag{4.27}$$

where $\boldsymbol{f}_{\text{in},i}$ and $\boldsymbol{m}_{\text{in},i}$ are inertial forces and moments exerted on link center of mass. Together with the following force and moment exerting on the joint or end-points,

$\boldsymbol{f}_i, \boldsymbol{m}_i$: force and moment on joint i

$\boldsymbol{f}_{e_i}, \boldsymbol{m}_{e_i}$: force and moment on end-point (if i is a terminal link)

the dynamic equilibrium for each link can be expressed in the following form:

$$\boldsymbol{f}_i = \boldsymbol{f}_{\text{in},i} + \sum_{j=i+1}^{n} S_{ij}\boldsymbol{f}_j + S_{e_i}\boldsymbol{f}_{e_i} \tag{4.28}$$

$$\begin{aligned}\boldsymbol{m}_i = \boldsymbol{m}_{\text{in},i} &+ \sum_{j=i+1}^{n} S_{ij}(\boldsymbol{\ell}_{ij} \wedge \boldsymbol{f}_j + \boldsymbol{m}_j) \\ &+ S_{ii}\boldsymbol{c}_{ii} \wedge \boldsymbol{f}_{\text{in},i} + S_{e_i}(\boldsymbol{\ell}_{ie} \wedge \boldsymbol{f}_{e_i} + \boldsymbol{m}_{e_i})\end{aligned} \tag{4.29}$$

for around a revolution joint (the only case considered in this thesis).

After the computation of whole $\boldsymbol{f}_i$ and $\boldsymbol{m}_i$ for $i = 1, ..., n$, joint torques can be obtained as:

$$\tau_i = \{m_i\}_I^T\{\hat{z}_i\} \tag{4.30}$$

(always for revolution joints).

The reaction force/moment on the base center of mass can be obtained as:

$$\boldsymbol{f}_0 = \sum_{i=1}^{n} S_{0i}\boldsymbol{f}_i \tag{4.31}$$

$$\boldsymbol{m}_0 = \sum_{i=1}^{n} S_{0j}(\boldsymbol{c}_{0i} \wedge \boldsymbol{f}_i + \boldsymbol{m}_i) \tag{4.32}$$

4.3.4 Forward dynamics

Simulating the UAM means running its forward dynamics. As seen in 3.3.3, the forward dynamics gives as output the accelerations of the generalized coordinates

($\dot{\mathbf{v}}_b$ and $\ddot{\boldsymbol{q}}$) of the system which then integrated give velocities and positions in the following instants. The inputs are all the forces acting on the system. Joint torques $\boldsymbol{\tau}$ may not be known if the manipulator is meant to be controlled kinematically in velocity or acceleration (as is done in 6). For this reason, before running the forward dynamics function implemented in MATLAB, inverse dynamics is performed, thus finding the torques that the motors deliver in order to follow the kinematic references given by control.

The procedure to compute the forward dynamics solution in discrete time as done in MATLAB simulator, is summarized as follows:

1. At time t, compute link positions and velocities, recursively from 0 to n.
2. Compute the inertia matrices using Equations (4.18)÷(4.23)
3. Set the accelerations $\ddot{\boldsymbol{x}}_b$, $\ddot{\boldsymbol{q}}$ and external forces $\boldsymbol{F}_b$, $\boldsymbol{F}_\mathrm{e}$ to zero, then compute the inertial forces recursively from link n to 0 using the recursive Newton-Euler function described in 4.3.3. the resultant forces on the coordinates $\boldsymbol{x}_b$ and $\boldsymbol{q}$ are equal to the non-linear forces $\boldsymbol{c}_b$ and $\boldsymbol{c}_m$, respectively.
4. Determine joint control forces $\boldsymbol{\tau}$ from a control law or by inverse dynamics if the manipulator is kinematically controlled, and base control forces $\boldsymbol{F}_b = \{\boldsymbol{U}_1, \boldsymbol{U}_2\}$ from UAV control law (2.6).
5. Compute the accelerations by
$$\begin{Bmatrix} \ddot{\boldsymbol{x}}_b \\ \ddot{\boldsymbol{q}} \end{Bmatrix} = \begin{bmatrix} [\boldsymbol{H}_b] & [\boldsymbol{H}_{bm}] \\ [\boldsymbol{H}_{bm}]^T & [\boldsymbol{H}_m] \end{bmatrix}^{-1} \left(\begin{Bmatrix} \boldsymbol{F}_b \\ \boldsymbol{\tau} \end{Bmatrix} + \begin{bmatrix} [\boldsymbol{J}_b]^T \\ [\boldsymbol{J}_m]^T \end{bmatrix} \boldsymbol{F}_e - \begin{Bmatrix} \boldsymbol{c}_b \\ \boldsymbol{c}_m \end{Bmatrix} \right) \tag{4.33}$$
In real systems servo-motors have an inner control loop dedicated to the tracking of input profiles, with their own dynamics. Usually, however, servo-motors performances are good enough to neglect the inner loop dynamics. Thus, in this work, the motors are assumed to track perfectly the reference velocity or acceleration profiles coming out from the control algorithm, and the forward dynamics output $\ddot{\boldsymbol{q}}$ is not be used.
6. Integrate the above accelerations to yield the velocities and positions at time $t + \Delta t$. The results reported in this work are obtained by using Euler

integration method, already mentioned in 3.2.2, for faster computations. The simulator however gives also the possibility to solve the dynamics differential equations through a Runge-Kutta method.

7. Go to 1. and continue

A note should be made about the integration (step 6.) of base attitude parameters rates of change. The usual routine for singularity-free integration from angular velocity to attitude involves the use of the relationship:

$$[\dot{\boldsymbol{R}}_{0I}] = [\boldsymbol{S}(\boldsymbol{\omega}_0)][\boldsymbol{R}_{0I}] \tag{4.34}$$

where $[\boldsymbol{S}(\boldsymbol{\omega}_0)]$ is the skew-symmetric form of the base angular velocity $\boldsymbol{\omega}_0$ and $[\boldsymbol{R}_{0I}]$ is the direction cosines matrix. However it was noted that, for numerical operations in practice, the following expression, which is known as Rodrigues formula at infinitesimal rotation, provides smaller error than (4.34):

$$[\boldsymbol{R}_{0I}(t+\Delta t)] = \left[[\mathbb{I}_3] + \sin\theta_0[\boldsymbol{S}(\boldsymbol{\omega}_0)] + (1-\cos\theta_0)[\boldsymbol{S}(\boldsymbol{\omega}_0)]^T[\boldsymbol{S}(\boldsymbol{\omega}_0)]\right][\boldsymbol{R}_{0I}(t)] \tag{4.35}$$

with

$$\theta_0 = |\boldsymbol{\omega}_0 \Delta t|$$

For this reason, this formula is used in the simulator instead of (4.34).

4.4 Validation of the model

As said, forward and inverse dynamics were implemented in MATLAB code with the objective of creating a fast simulation environment for UAMs where to test the different control algorithms that are presented in Chapter 6. ADAMS software, one of the leading computational codes for multibody simulations, was used as a benchmark to validate it. Figure 4.5 shows a picture of the simplified UAM model that was used in ADAMS.

As benchmark virtual experiment, UAM manipulator was requested to track a circular trajectory in x-z plane. Other experiment characteristics are specified in Table 4.1.

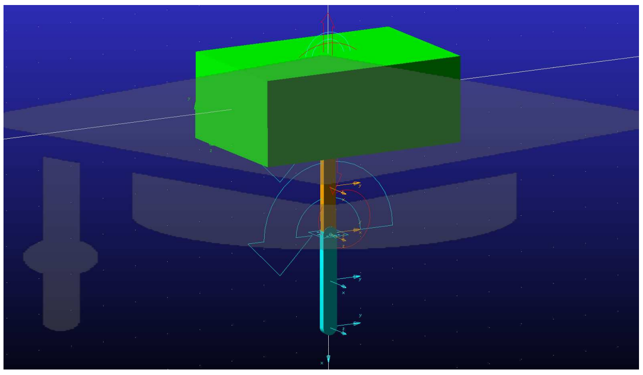


Figure 4.5: ADAMS model

Circular trajectory diameter (D)	0.08 m
Trajectory travel time T_{path}	6 s
Time after reaching trajectory final point T_{post}	3 s

Table 4.1: Benchmark experiment characteristics

Motors reference velocities $\dot{\boldsymbol{q}}$ were planned in MATLAB through the inverse kinematics algorithm that is presented in Chapter 6, and given as input both to MATLAB and ADAMS models. The results from the two environments are then compared in order to see MATLAB code accuracy. In particular, it was chosen to compare the temporal measures of the end-effector x and z (x_{ee}, z_{ee}) coordinates and to see the distance between the trajectories.

Figure 4.6 and 4.7 show the graphical output of the two simulation engines after the end of the virtual experiments.

Figure 4.8 shows x_{ee} and z_{ee} temporal measure both for the ADAMS and the MATLAB experiments. Figure 4.9 shows the plot of the difference between EE x and z coordinates during the experiments $(x_{ee_{\text{MATLAB}}} - x_{ee_{\text{ADAMS}}}, z_{ee_{\text{MATLAB}}} - z_{ee_{\text{ADAMS}}})$. Maximum differences are smaller than 1 mm indicating that the MATLAB simulator is sufficiently reliable.

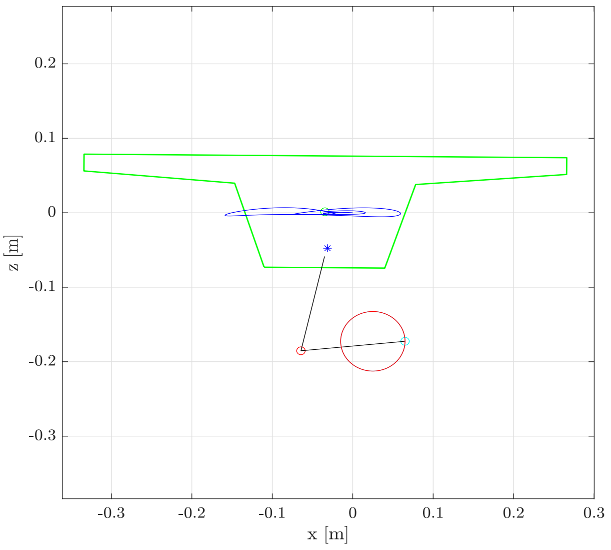


Figure 4.6: MATLAB simulator graphical output. In red the travelled end-effector trajectory and in blue G_{UAV} trajectory

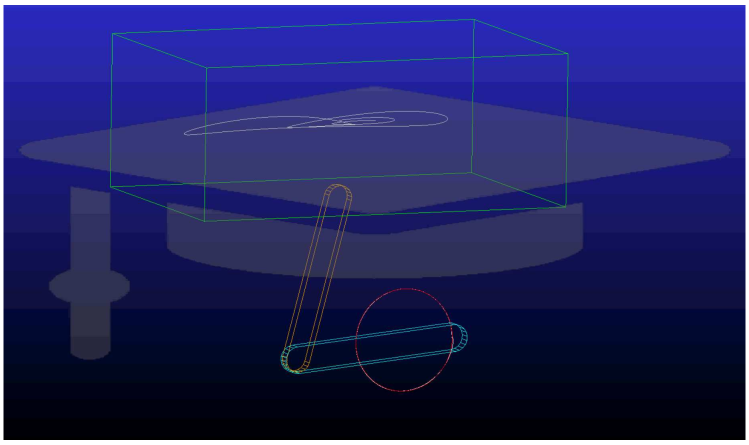


Figure 4.7: ADAMS graphical output

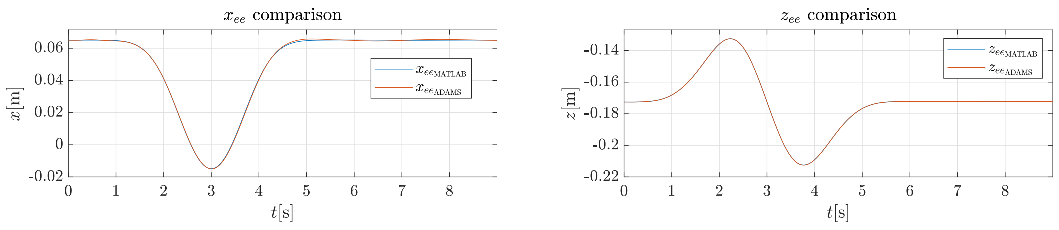


Figure 4.8: End-effector x and z coordinates in the two experiments

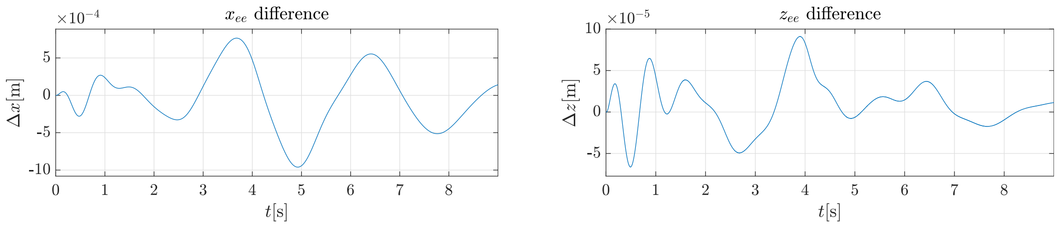


Figure 4.9: Differences between end-effector x and z coordinates

Chapter 5

Disturbances models

As mentioned, one of the main purposes of the simulation environment is to quickly test the UAM control algorithms performances under different conditions. Therefore, the simulator has been enriched by implementing models of the disturbances that are expected to affect the most aerial manipulators in their work.. The goal of this chapter is to describe how these disturbances have been modelled in MATLAB.

Three main causes of disturbance to aerial manipulators operations have been identified:

- Disturbances of aerodynamic nature.
- Disturbances coming from noise in the measurement chain.
- Uncertainty on the parameters of the model on which the control system is based.

The first category of disturbances turns out to be very difficult to model accurately. Most of the models that can be found in literature are based on empirical formulas which coefficients come either from data derived from expensive experiments on UAVs, or from sophisticated CFD simulations. The main objective of this thesis is still to test the rejection capabilities of UAMs control systems. For this purpose the aerodynamic model does not need to be too realistic. So a simple model of aerodynamic disturbance consisting of a concentrated force acting on the UAV's center of gravity was chosen. In section 5.1 this force represents a disturbance coming from wind gusts.

Disturbances belonging to the second category are perhaps the most impactful when dealing with UAVs. As seen these platforms need feedback controllers in order to function properly. Feedback control loops need informations on the actual system status which come from sensors like those mentioned in 2.4.1. Unfortunately in practice, those informations are often affected by noise. This may compromise the performances of the controller, if not make it unstable. In order to test the reliability of the different control schemes under this condition, in section 5.2 a Gaussian distributed noise model was implemented and applied to some quantities that will have to be measured in UAMs practical applications.

The control schemes that will be presented in Chapter 6 are heavily based on UAM model. An uncertainty about the knowledge of some model parameters may bring the system not to work as expected or, worse, to instability. Since, in practice, many of the model parameters may not be known with great confidence, it was thought to test the control schemes robustness by varying the model parameters in the dynamic simulation. Which parameters and how they were modified is described in Section 5.3.

5.1 Casual wind

UAMs applications may be outdoor or indoor. In outdoor applications the most important external disturbance that will hinder the proper tracking of a trajectory by the manipulator is wind. In indoor applications the wind may not be felt but still there are aerodynamic disturbances causing aircraft oscillations (see the results of the experiment in [5]). These, in this work, will be likened to a force caused by a fictitious wind. Therefore, a wind model is needed in order to simulate the disturbances in both the scenarios.

This section will describe the two models that have been implemented for the aerodynamic force on the UAV. The first, called Dryden model, is for generating random wind speed profiles. The second, taken from [15], is used to match a wind speed profile with a force on the aircraft.

5.1.1 Dryden model

The wind velocity vector can be characterized as the sum of two vectors:

$$\boldsymbol{w} = \boldsymbol{w}_0 + \Delta\boldsymbol{w}(t) \tag{5.1}$$

- The static wind vector $\boldsymbol{w}_0$ evolves slowly along time, it represents the mean value of the overall wind vector. This component will be ignored in this work.
- The fast-time varying or gust component $\Delta\boldsymbol{w}(t)$ features higher frequencies and smaller amplitudes in comparison with the static wind vector. Since the first component will be neglected it will be $\boldsymbol{w} = \Delta\boldsymbol{w}(t)$.

The Dryden model is a wind gust model dependent on a standard Power Spectral Density (PSD). The parameters of the PSD may be derived from static wind data for realistic results or set to create profiles with certain characteristics (for example high frequency and low amplitude profiles for simulating disturbances in indoor environments). The reference for this model is [16]. One wind velocity vector component is defined as a sum of sinusoidal components:

$$w(t) = \sum_{i=1}^{n} a_i \sin(\Omega_i t + \varphi_i) \tag{5.2}$$

where Ω_i and φ_i are randomly selected frequencies and phase shifts, n is the number of sinusoids and a_i are the amplitude of sinusoids. For simplicity the frequencies were chosen as to divide in equal parts the interval they are picked from. For realistic results in [16] it is suggested to take values of Ω_i between 0.1 and 1.5 rad/s. The magnitudes a_i are given by

$$a_i = \sqrt{\Delta\Omega_i \Phi(\Omega_i)} \tag{5.3}$$

where $\Delta\Omega_i$ are the intervals between the frequencies and $\Phi(\Omega_i)$ (all equally long for the assumption made) and $\Phi(\Omega_i)$ is the power spectral density function evaluated for $\Omega = \Omega_i$. The PSD for horizontal and vertical wind are different and are found from the following equations:

$$\Phi_h(\Omega) = \sigma_h^2 \frac{2L_h}{\pi} \frac{1}{1 + (L_h\Omega)^2} \tag{5.4}$$

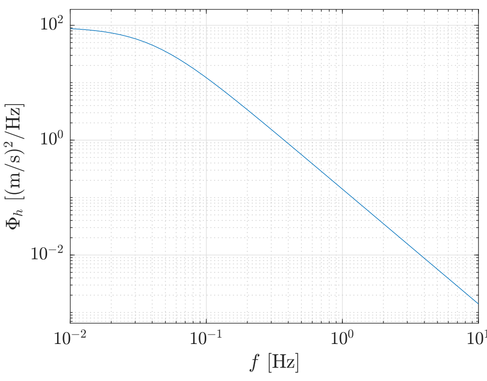
$$\Phi_v(\Omega) = \sigma_v^2 \frac{L_v}{\pi} \frac{1 + 3(L_v\Omega)^2}{(1 + (L_v\Omega)^2)^2} \tag{5.5}$$

Here, σ_h and σ_v are the horizontal and vertical turbulence intensities, L_h and L_v are the horizontal and vertical gust length scale. The vertical length scale and turbulence intensity can be assumed to be $L_v = |z|$ (with z being the vehicle altitude) and $\sigma_v = 0.1\omega_{20}$ (ω_{20} is the given wind speed in knots at 20ft altitude which in [17] is said to be 15 knots for light turbulence). The altitude was chosen to be $z = 1\text{m} = 3.28\text{ft}$. Finally L_h and σ_h can be found with the following equations:

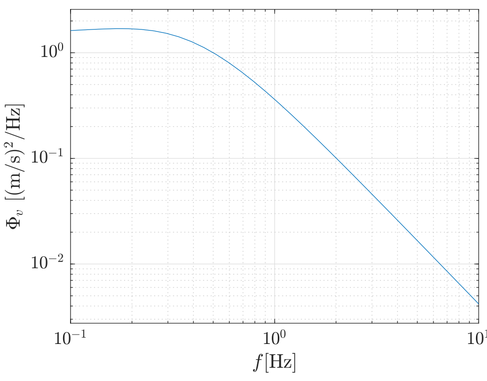
$$\frac{L_h}{L_v} = \frac{1}{(0.177 + 0.000823z)^{1.2}} \tag{5.6}$$

$$\frac{\sigma_h}{\sigma_v} = \frac{1}{(0.177 + 0.000823z)^{0.4}} \tag{5.7}$$

The wind field model therefore defines the temporally varying wind speed at a given altitude. In Figure 5.1 are plotted the PSDs for horizontal and vertical wind with the above mentioned parameters. Figure 5.2 shows random horizontal ($w_x(t)$) and vertical ($w_z(t)$) wind profiles.



(a) Horizontal wind PSD



(b) Vertical wind PSD

Figure 5.1: Power Spectral Density functions log-log plots

5.1.2 Aerodynamic force on the UAV

In order to rely a wind velocity vector to a force acting on the UAV, the model described in [15] (called Solovyev model for its author, in this work) was adopted. This model has the advantage of being simple, compared to other that can be

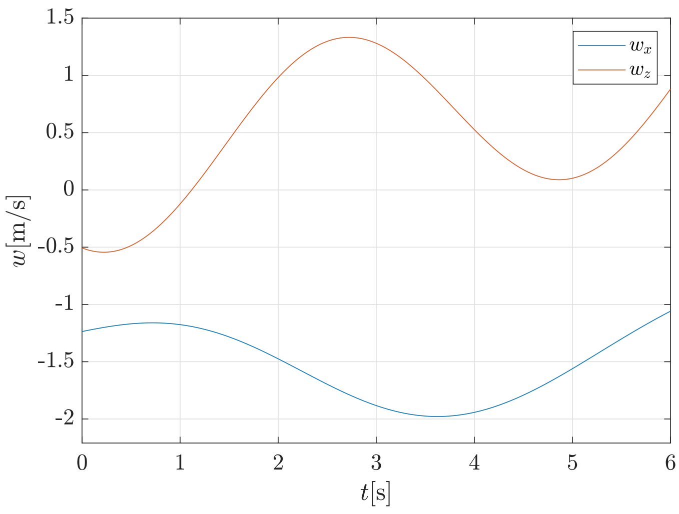


Figure 5.2: Example of wind velocity profile. $n = 40$ and $\Omega_i \in [0.1, 1.5]$ rad/s

found in literature, and therefore easily implementable. Given the wind velocity, the model defines the force exerted on the UAV with the formula:

$$F_w = S_e A w^2. \tag{5.8}$$

S_e is the influence effective areas and $A = 1/16 \cdot 9.81$ is the rate of wind velocity [m/s^2] conversion to pressure [15]. For simulation tasks it is handier to use the components of formula (5.8) as projection on the axes of the inertial frame

$$F_{w_x} = S_{e_x} A(w_x^2(t) + w_z^2(t)) \cos \psi_w \tag{5.9}$$

$$F_{w_z} = S_{e_z} A(w_x^2(t) + w_z^2(t)) \sin \psi_w \tag{5.10}$$

where ψ_w is the inclination angle of the wind velocity vector with respect to the horizontal.

For simplicity sake the UAV is represented by a cylindrical surface. It's surface is given by the sum of the area S_l of the lateral surface and the area S_b of the bases. The expression for UAV surface area is therefore:

$$S = S_l + S_b = \alpha 2\pi r h + \beta 2\pi r^2 \tag{5.11}$$

where r is the cylinder radius, h is the height and α and β are fill factors for the cylinder areas (depending on UAV design). The wind is assumed to uniformly influence half of the cylindrical surface like in Figure 5.3.

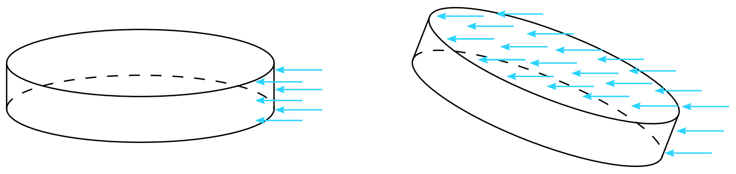


Figure 5.3: Wind effect on the cylindrical surfaces

Taking into account formula (5.11), the expression for the effective area is:

$$S_{e_x} = \beta\pi r^2 \sin\theta + \alpha\pi r h \cos\theta, \; S_{e_z} = \beta\pi r^2 \sin\phi + \alpha\pi r h \cos\phi \tag{5.12}$$

where ϕ and θ and are the roll and pitch angles respectively. Since in this work the roll angle will be ignored and the pitch angle will stay small (UAV near hovering conditions), S_{e_z} will be assumed to stay constant.

Figure 5.4 shows the force acting on the UAV corresponding to the wind velocity profiles of Figure 5.2. Solovyev model parameters are enlisted in Table 5.1.

Cylinder radius r	454 mm
Cylinder height h	175 mm
Base area fill factor α	0.2736
Lateral area fill factor β	0.2768

Table 5.1: UAV parameters for Solovyev model

5.2 Measurements noise

In practice the controller will have to deal with noisy signals coming from the different sensors which the UAM is equipped with. To test the controller's ability to deal with real signals, Gaussian distributed noise will be added to some kinematic quantities in output from the simulator's dynamics block. The noisy signal can be represented as:

$$\mathbf{X} = x + \sigma \cdot \mathbf{Z} \quad , \quad \mathbf{Z} \sim \mathcal{N}(0,1) \tag{5.13}$$

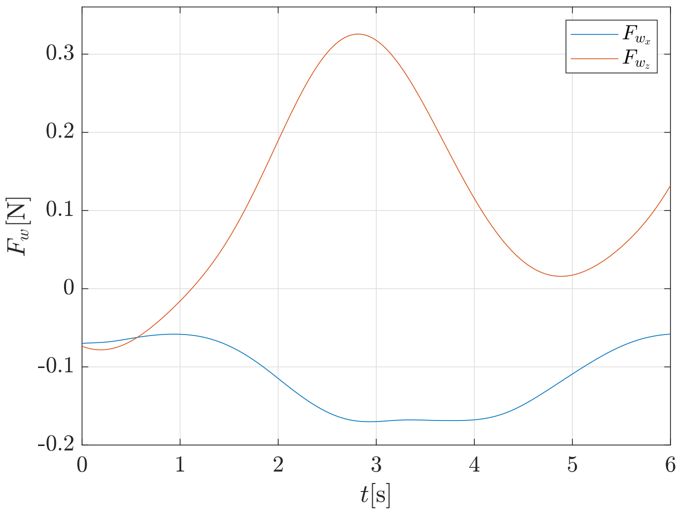


Figure 5.4: Aerodynamic forces on the UAV for the wind velocity profiles in Fig. 5.2

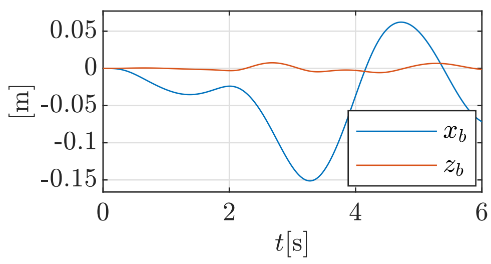
where $x = x(t)$ symbolizes the real signal, σ is the standard deviation and depends on the characteristic of the specific sensor used for measuring the signal, and $\mathbf{Z}$ is a random Gaussian variable. $\mathcal{N}(0, 1)$ is the standard Gaussian probability density function (with mean $\mu = 0$ and variance $\sigma^2 = 1$). In order to generate a random Gaussian variable in MATLAB it is used the built-in function `randn`.

The variables to which Gaussian noise was added are:

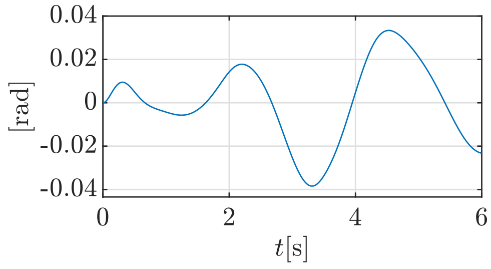
- Measure of UAV center of mass displacements $\boldsymbol{x}_b$ and velocity $\boldsymbol{v}_b$
- Measure of UAV pitch angle θ
- Measure of UAV angular velocity ω_b

As stated in 2.4.1 these variables may be measured by different sensors like cameras, motion capture systems, IMU etc... each with its own characteristics in terms of noise parameters. Thus, σ would be different depending on which sensor is used.

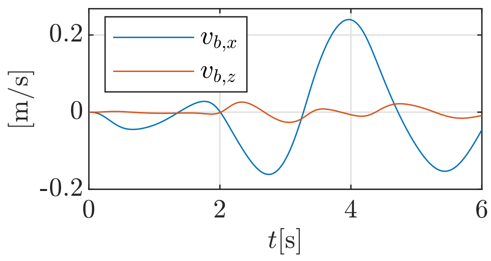
Figure 5.5 shows the plots of the three above mentioned quantities in the tracking of a circular trajectory (specifics like in 4.1 except for $T_{\text{post}} = 1\text{s}$) when no noise is implemented. The velocities of the arm motors were calculated using the RMRC algorithm explained in 6.2.



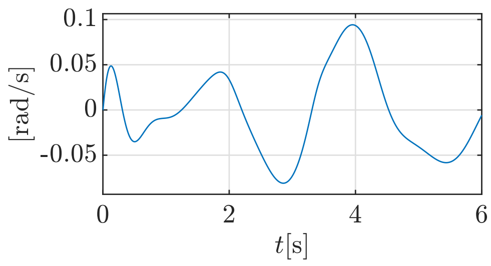
(a) UAV translations



(b) UAV rotations



(c) UAV linear velocities



(d) UAV angular velocity

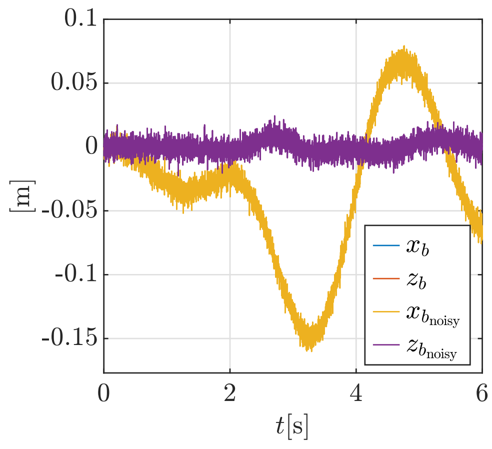
Figure 5.5: UAV kinematic quantities outputs of the dynamics block plots

Figure 5.6 shows how kinematic quantities signals change if Gaussian noise is added like in (5.13). In Table 5.2 are reported the standard deviations used for each signal.

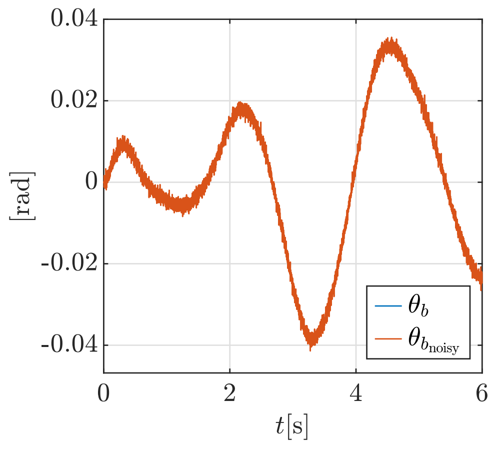
Signal	σ
Base position $\{x_b, y_b\}$	0.005 m
Base velocities $\{v_{b,x}, v_{b,z}\}$	0.005 m/s
Base pitch angle θ_b	0.001 rad
Base angular velocity ω_b	0.005 rad/s

Table 5.2: Standard deviations used for modeling Gaussian noise to each signal

The behaviour of the system is different in this case since the velocities of the arm motors are the same, but the UAV controls slightly change due to the noisy inputs. Figure 5.7 shows the plots of the UAV control forces U_1, U_2 in the case of noisy inputs.

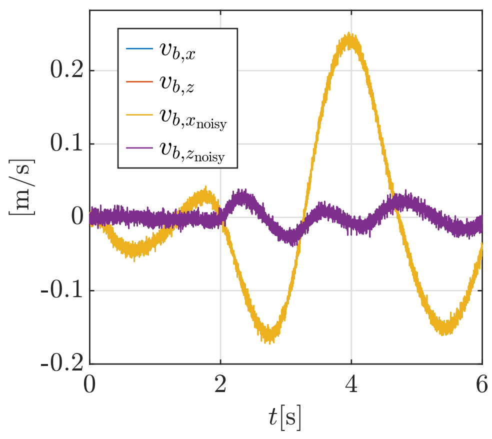


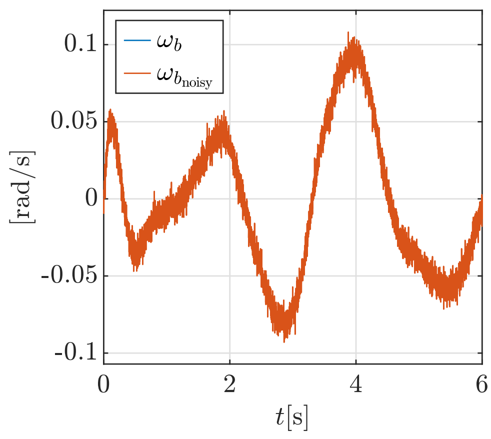
(a) UAV translations



(b) UAV rotations

(c) UAV linear velocities





(d) UAV angular velocity

Figure 5.6: UAV kinematic quantities outputs of the dynamics block plots corrupted by noise

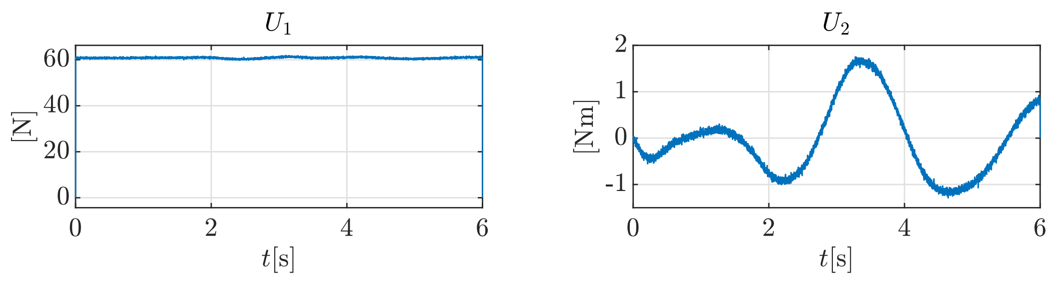


Figure 5.7: UAV control actions in the case of noisy inputs

In this work were implemented control algorithms based on the operational space error like those described in 3.4. For these to work the system needs the information about end-effector position and maybe velocity. This can be obtained through a sensor or through direct kinematics. In both the cases end-effector position information will result noisy, or due to the sensor itself or due to the propagation of the noise corrupting the other signals. Figure 5.8 shows how the noise on the UAV kinematic quantities can influence the accuracy with which the end-effector position is known. This is computed in the simulator through direct kinematics function. No additional noise is added to the measure.

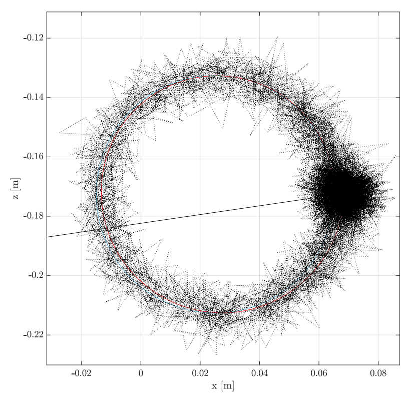


Figure 5.8: End-effector positions for direct kinematics. In black the result of direct kinematics. Underneath, in red, the real position

5.3 Parameters uncertainty model

In order to to test the robustness of the control algorithms, it is important to be able to verify how they behave when the parameters of the real system differ from those of the model. For implementing this uncertainty, in the simulator script, the inverse kinematics and forward dynamics are carried out separately in two blocks, one before the other. In this way, by setting different parameters before the dynamics block, it is possible to solve the direct dynamics for an UAM characterized by different parameters than the one to which the inverse kinematics had referred. Three categories of parameters were identified on which there might be greater uncertainty:

- Uncertainty on the inertial parameters of both the UAV and the manipulator links (m_{UAV}, $[\boldsymbol{I}_{G_b}]$, m_i, $[\boldsymbol{I}_{G_i}]$).
- Lack of accuracy on the knowledge of the location of the barycenters of the links and the displacement of the first joint relative to the drone's barycenter.
- Complete or partial ignorance about the mass of the load m_{load} to be carried by the manipulator.

5.3.1 Uncertainty on the inertial parameters

UAV mass and inertia parameters may be known with poor accuracy, not only because of measurement errors, but also because of the often variable equipment with which they are operated. Different missions might require different instruments to carry. Even the same sensors needed for the control the UAV might change. These changes in the aircraft load may not be matched by the necessary changes in the model used by the control. It is necessary for the control system to be robust to the variation of these parameters given the high probability of these situations occurring.

To virtually reproduce this type of situations, in the simulator, before dynamic block, the parameters of the UAV are modified as:

$$m_{\text{UAV,unc}} = m_{\text{UAV}} + r_{[-1,1]} u_{\max} m_{\text{UAV}} \quad , \quad I_{G_b,\text{unc}} = I_{G_b} + r_{[-1,1]} u_{\max} I_{G_b} \tag{5.14}$$

where:

- m_{UAV} and $m_{\text{UAV,unc}}$ are the real and model UAV masses
- $I_{G_b,\text{unc}}$ and I_{G_b} are the real and model UAV moments of inertia (with respect to G_b)
- $u_{\max}$ is the maximum uncertainty that the model can have. It's set to 0.15 (uncertainty of maximum 15% of the real value for that parameter)

- $r_{[-1,1]}$ is a random number between -1 an 1. It's generated in MATLAB through the construct `(2*rand-1)`. `rand` gives a random number between 0 and 1 picked from a uniform distribution.

Link parameters are modified similarly as:

$$m_i = m_i + r_{[-1,1]} \cdot u_{\max} \cdot m_i \quad , \quad I_{G_i} = I_{G_i} + r_{[-1,1]} \cdot u_{\max} \cdot I_{G_i} \tag{5.15}$$

The uncertainty $u_{\max}$ is 2% for the link masses and 5% for the moments of inertia.

5.3.2 Barycenters locations uncertainty

In order to simulate an uncertainty on the UAV CoG position $\boldsymbol{r}_b$, the displacement of manipulator first joint with respect to G_b is slightly changed. Let $\Delta x_{G_b,j1}$ and $\Delta z_{G_b,j1}$ be the coordinates of first joint relative positions vector in the inertial frame. These are redefined as:

$$\Delta x_{G_b,j1} = r_{[-1,1]} \cdot \Delta x_{\max} \quad , \quad \Delta z_{G_b,j1} = r_{[-1,1]} \cdot \Delta z_{\max} \tag{5.16}$$

Uncertainty is expected to be on the order of centimeters at most so $\Delta x_{\max}$, $\Delta z_{\max}$ are chosen equal to 2 cm.

The uncertainty about the position of the link's center of mass is assumed to be only along the direction joining its two joints. In order to model this it is sufficient to change the parameter $\boldsymbol{c}_{i,i}$. This is treated like the links inertial parameters with an uncertainty of maximum 2%. It should be noted that once a simulation is started UAM geometry is set.

5.3.3 Load uncertainty

As described in Chapter 4, the dynamic model of the UAM includes the possibility that a force is applied to the end organ of the manipulator. One way to simulate the holding of a load by the manipulator is to model this end organ force as the sum of the weight force and the inertia force of the load. In real world applications of UAMs for manipulation or grasping it may be that the load to be carried is known with great uncertainty, if not unknown. Tests of the acceleration control algorithm in 7.3.3 will be performed assuming the mass of the load to be carried (m_{load}) to be completely unknown.

Chapter 6

UAMs control

This chapter explains the control algorithms implemented in MATLAB for controlling the UAM. These algorithms are logically derived from CLIK (Closed-Loop Inverse Kinematics) schemes, such as those described in 3.4 for fixed-base manipulators. As for these, first-order (Resolved Motion Rate Control) and second-order (Resolved Acceleration Control) algorithms are distinguished. The former involve controlling the manipulator at the rate level, the latter at the acceleration level. Once given as input the desired end-effector trajectory in terms of velocity/acceleration, they give as output the velocity/acceleration profiles that are the references for the servo motors of the manipulator.

For each of the algorithms, the outcomes of two simulations are reported: one in which only forces that are assumed to be known are acting, and one in which an unpredicted triangular-pulse horizontal force acts halfway through the trajectory. The quantity that was chosen as the yardstick for each virtual experiment is the norm of end-effector position error $||\boldsymbol{e}(t)||$. All the specifics of the simulations performed are given in the next section.

6.1 Specifics of the simulations

In the virtual experiments to test the control algorithms, the end-effector of the aerial manipulator is required to follow a circular trajectory of diameter $D = 0.08$ m in a time $T_{\text{path}} = 5$ s with an arc length function $s(t)$ defined by:

$$\begin{cases} s(t) = \frac{C/2}{(T_{\text{path}}/2)^2}\left(\frac{t^2}{2} + (2\beta)^{-2}\cdot\cos(2\beta\ t) - (2\beta)^{-2}\right) & \text{if} \quad t < T_{\text{path}}/2 \\ s(t) = C - \left(2\frac{C/2}{(T_{\text{path}}/2)^2}\cdot(\frac{(T_{\text{path}}-t)^2}{2} + \right. & \\ \left. (2\beta)^{-2}\cdot\cos((2\beta)\cdot(\frac{T_{\text{path}}}{2} - t)) - (2\beta)^{-2})\right) & \text{if} \quad t \geq T_{\text{path}}/2 \end{cases}$$

with $C = 2\pi r = 0.25$ m being the circumference of the trajectory and $\beta = 2\pi/T_{\text{path}}$. $s(t)$ plot is shown in Figure 6.1.

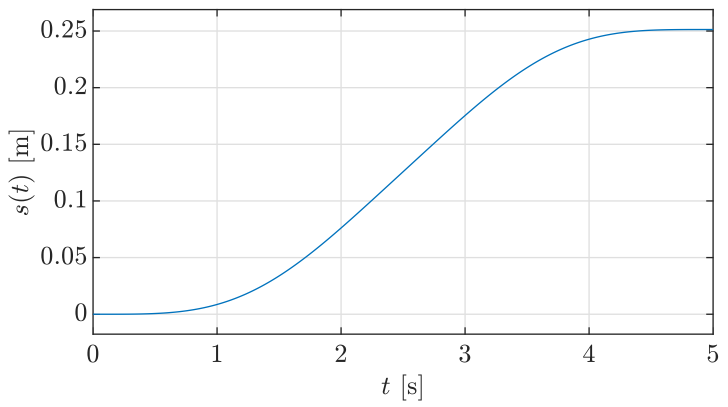


Figure 6.1: Arc length vs. time function of the circular trajectory to be followed by the UAM in the virtual experiments

After the end-effector has reached the final point, the simulation continues again for a time $T_{\text{post}} = 1$ s. This last period was added to be able to verify the ability of the control system to eliminate the steady-state error.

The trajectory starts from the end-effector initial position set from manipulator initial configuration. In all the simulations manipulator starts in the initial configuration $\{q_{1,\text{in}}, q_{2,\text{in}}\} = \{-\pi/6, 2\pi/3\}$ rad and drone starts with its center of mass at point $\{0, 0\}$ of the inertial frame.

The two main components of the UAM considered in the simulations are:

- The DJI S1000 drone described in Chapter 2 as base
- 2 DoFs planar manipulator. The links were modelled as one-dimensional and are 0.13 m long. The barycenters are assumed to be located exactly in the middle at a distance $c_{i,i}$ from the joint connecting it to the previous link. The first joint is assumed to be exactly under UAV center of mass at a distance $\Delta z_{G_b,j1} = 0.06$ m.

The geometric and inertial data of the UAM are summarized in Table 6.1. Refer to Figure 6.2 for the adopted symbology.

Parameter	**Value**
Base mass m_{UAV}	4.2 kg
Base moment of inertia I_{G_b}	0.4097 kg m^2
Manipulator first joint location $\{\Delta x_{G_b,j1}, \Delta z_{G_b,j1}\}$	$\{0, -0.06\}$ m
Links length ℓ_i	0.13 m
Link barycenter distance from previous joint $c_{i,i}$	0.065 m
Links moment of inertia I_{G_i}	$4{\cdot}10^{-4}$ kg m^2

Table 6.1: Geometric and inertial parameters of the reference UAM for the simulations

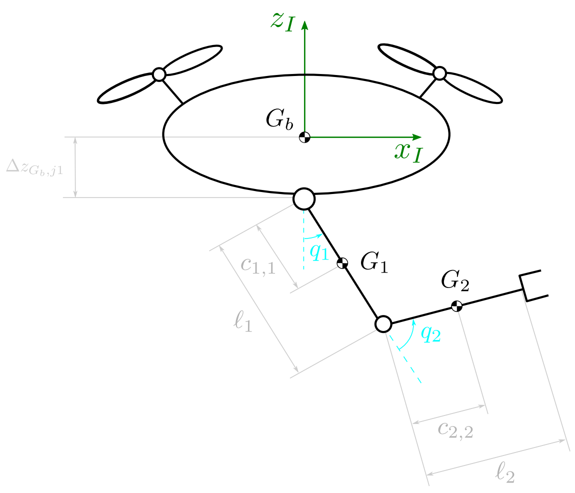


Figure 6.2: Schematics of the UAM used in the simulations

All the algorithmic solutions has been implemented in discrete time using the Euler integration rule (see 3.2.2) with 0.001 s time interval.

As mentioned, in this chapter, in order to test the algorithms, the UAM system will be subjected to the action of a triangular pulse force. The pulse starts at the instant $t = T_{\text{path}}/2$, arrives in 0.04 s at 10 N, and ends 0.04 s later. Its plot is shown in Figure 6.3.

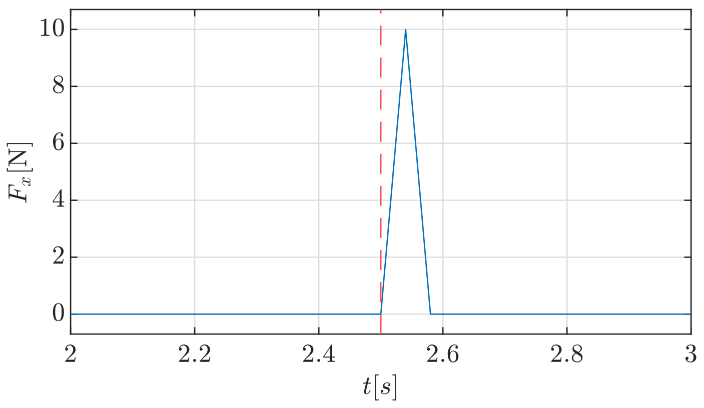


Figure 6.3: Plot of the triangular-pulse horizontal force acting on the UAV at $t = T_{\text{path}}/2$ (red dotted line) in the simulations.

In all the simulations the UAM base is subjected to to the weight force and the forces $\boldsymbol{U}_1$, $\boldsymbol{U}_2$, exerted by the propulsion system, determined through the PID control reported in Equation 2.6. The goal of PID control is to return the UAV to the hovering condition in the state $\{z, \theta\} = \{0, 0\}$. It should be noted that the motion along x of the drone is not controlled.

6.2 Resolved Motion Rate Control (RMRC)

Resolved Motion Rate Control (RMRC) is a method for continuous path control of a manipulator hand based on the inversion of Jacobian matrix. In this section is presented an RMRC algorithm for aerial manipulators that builds on the work developed in [18],[19] for space manipulators and extended in [20] for UAMs.

Consider Equation (4.24). This can be solved for velocities yielding

$$\dot{\boldsymbol{x}}_b = \begin{Bmatrix} \boldsymbol{v}_b \\ \dot{\boldsymbol{\phi}}_b \end{Bmatrix} = -[\boldsymbol{H}_b]^{-1}[\boldsymbol{H}_{bm}]\dot{\boldsymbol{q}} + [\boldsymbol{H}_b]^{-1} \begin{Bmatrix} \boldsymbol{p} \\ \boldsymbol{K} \end{Bmatrix}. \tag{6.1}$$

In light of 6.1 the kinematic equation (4.13) can be rewritten as

$$\begin{aligned}
\dot{\boldsymbol{x}}_e &= [\boldsymbol{J}_b]\dot{\boldsymbol{x}}_b + [\boldsymbol{J}_m]\dot{\boldsymbol{q}} \\
&= [\boldsymbol{J}_b]\left(-[\boldsymbol{H}_b]^{-1}[\boldsymbol{H}_{bm}]\dot{\boldsymbol{q}} + [\boldsymbol{H}_b]^{-1}\begin{Bmatrix}\boldsymbol{p}\\ \boldsymbol{K}\end{Bmatrix}\right) + [\boldsymbol{J}_m]\dot{\boldsymbol{q}} \\
&= \left([\boldsymbol{J}_m] - [\boldsymbol{J}_b][\boldsymbol{H}_b]^{-1}[\boldsymbol{H}_{bm}]\right)\dot{\boldsymbol{q}} + [\boldsymbol{J}_b][\boldsymbol{H}_b]^{-1}\begin{Bmatrix}\boldsymbol{p}\\ \boldsymbol{K}\end{Bmatrix} \\
&= [\boldsymbol{J}_{\text{gen}}]\dot{\boldsymbol{q}} + [\boldsymbol{J}_b][\boldsymbol{H}_b]^{-1}\begin{Bmatrix}\boldsymbol{p}\\ \boldsymbol{K}\end{Bmatrix}
\end{aligned} \tag{6.2}$$

where $[\boldsymbol{J}_{\text{gen}}]$ is called Generalized Jacobian Matrix. The origin of this name comes from the fact that, if the linear and angular momenta are null, Equation (6.2) is formally analogous to Equation (3.5) for fixed-base manipulators. The assumption of zero moments $\boldsymbol{p}$ and $\boldsymbol{K}$ is very common in the case of space manipulators. Since in space the net external force on the system is usually zero, $\boldsymbol{p}$ and $\boldsymbol{K}$ are conserved. If these are assumed to start with initially zero $\boldsymbol{p}$ and $\boldsymbol{K}$, they are conserved and stay null.

This is not the case with aerial manipulators though. These are subject to the weight force, the forces exerted by the propulsion system and other disturbance forces. In this case $\boldsymbol{p}$ and $\boldsymbol{K}$ may be zero initially, but then they must be updated since their contribution is not negligible as can be seen from equation 6.2.

Given the external force $\boldsymbol{F}_e = \{\boldsymbol{f}_e, \boldsymbol{m}_e\}^T$ acting on the system, the algorithm updates at each timestep the linear and angular moments as:

$$\boldsymbol{p}(t_{k+1}) = \boldsymbol{p}(t_k) + \boldsymbol{f}_e(t_k)\cdot\Delta t \tag{6.3}$$

$$\boldsymbol{K}(t_{k+1}) = \boldsymbol{K}(t_{k+1}) + \left(-\boldsymbol{v}_b(t_k)\wedge\boldsymbol{p}(t_k) + \boldsymbol{m}_e\right)\Delta t \tag{6.4}$$

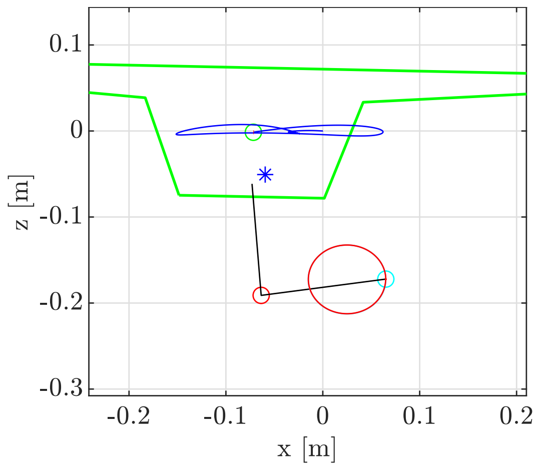
In this work, only the weight force and the UAV control forces U_1 and U_2 are considered as known external loads for $\boldsymbol{f}_e$ and $\boldsymbol{m}_e$. The reader can refer to [5] and [20] for more details on the numerical procedure.

From Equation (6.2) it is possible to derive the velocities that the manipulator joints must have, once the desired velocity $\boldsymbol{x}_{e,\text{des}}$ of the end-effector is assigned, as:

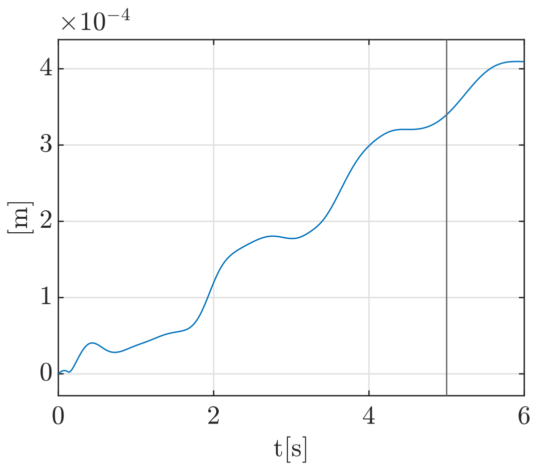
$$\dot{\boldsymbol{q}} = [\boldsymbol{J}_{\text{gen}}]^{-1}\left(\dot{\boldsymbol{x}}_{e,\text{des}} - [\boldsymbol{J}_b][\boldsymbol{H}_b]^{-1}\begin{Bmatrix}\boldsymbol{p}\\ \boldsymbol{K}\end{Bmatrix}\right) \tag{6.5}$$

Equation 6.5 represents the formula for solving inverse kinematics at the velocity level for the arm of a UAM. If measurements are not available, once $\dot{\boldsymbol{q}}$ is known, the base velocities can be derived from Equation (6.1).

In the case where the forces acting on the UAV or end-effector are all known, formula (6.5) shows how a kinematic-type control can be implemented for trajectory tracking in operational space. This control could work in the case where only predictable forces that can be directly computed by the controller or that can be estimated by on-board sensors data act on the system. Figure 6.4 shows that the UAM is indeed able to follow the circular trajectory described in 6.1 if the velocities dictated by (6.5) are imparted to the motors. The only forces acting in this case are the weight force and the controls U_1 and U_2, which are known and updated in the controller model to calculate $\boldsymbol{p}$ and $\boldsymbol{K}$.



(a) EE trajectory (cyan=desired, red=actual) and G_b trajectory (in blue)



(b) norm of the EE position error over time $||\boldsymbol{e}(t)||$, black vertical line at $t = T_{\text{path}}$

Figure 6.4: Trajectory tracking with motors velocities imparted by (6.5). The only forces acting on the system are the weight force, U_1 and U_2 which are known and updated in the controller

In the case where the forces acting on the system cannot be estimated with sufficient accuracy or cannot be estimated at all, (6.5) would not provide suitable velocities for tracking the desired trajectory. To effectively control the system, a feedback term, based on the operational space error $\boldsymbol{e} = \boldsymbol{x}_{e,\text{des}} - \boldsymbol{x}_e$, must be added to the velocity references calculated through (6.5). A CLIK algorithm, such as those described in 3.4, is thus obtained.

The new velocities $\dot{\boldsymbol{q}}$ assigned by the algorithm can then be written as:

$$\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_{\text{ref,ik}} + \Delta\dot{\boldsymbol{q}}_{\text{feedback}} = \dot{\boldsymbol{q}}_{\text{ref}} + \Delta\dot{\boldsymbol{q}} \tag{6.6}$$

where $\dot{\boldsymbol{q}}_{\text{ref,ik}}$ is the velocity vector calculated through (6.5) and represents a feed-forward term while $\Delta\dot{\boldsymbol{q}}_{\text{feedback}}$ represents the feedback term. In the simulator, the velocities $\dot{\boldsymbol{q}}_{\text{ref,ik}}$ are calculated a priori via (6.5) in a kinematics block where the moments are updated under the assumption that only the weight force and U_1, U_2 are acting. To these are then added the $\Delta\dot{\boldsymbol{q}}_{\text{feedback}}$ velocities that are computed in the block where the UAM direct dynamic simulation takes place (see 4.3.4).

Taking inspiration from 3.4.1 (Jacobian inverse control) it is possible to choose the feedback term as:

$$\Delta\dot{\boldsymbol{q}} = [\boldsymbol{J}_{\text{gen}}]^{-1}[\boldsymbol{K}_P]\boldsymbol{e} \tag{6.7}$$

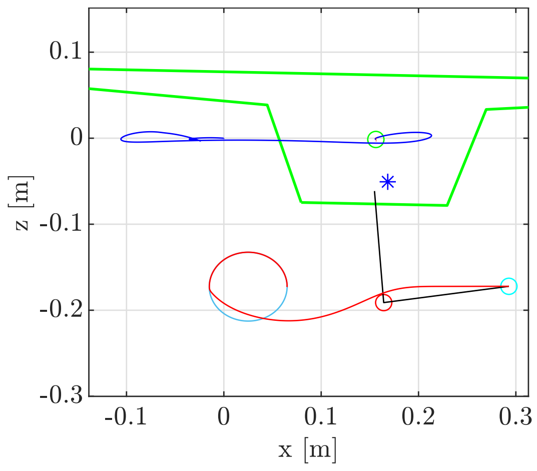
where $[\boldsymbol{K}_P] = K_P[\mathbb{I}_2]$ is diagonal and positive definite matrix of gains. Substituting (6.7) into (6.6) and taking into account Equation (6.5), $\dot{\boldsymbol{q}}$ can be written as

$$\dot{\boldsymbol{q}} = [\boldsymbol{J}_{\text{gen}}]^{-1}\left(\dot{\boldsymbol{x}}_{e,\text{des}} - [\boldsymbol{J}_b][\boldsymbol{H}_b]^{-1}\begin{Bmatrix}\boldsymbol{p}\\ \boldsymbol{K}\end{Bmatrix} + [\boldsymbol{K}_P]\boldsymbol{e}\right) \tag{6.8}$$

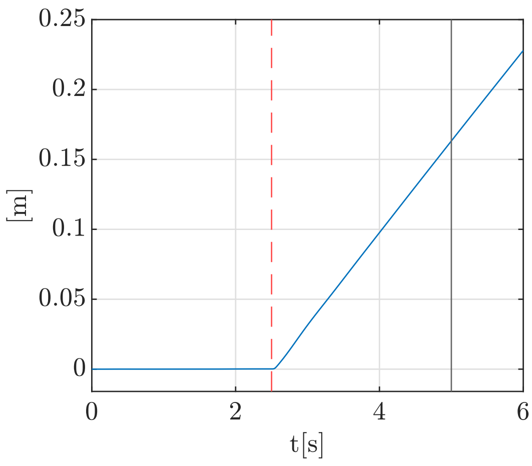
To know the error, it is necessary to know the current position of the end-effector so that it can then be compared with the desired position. In the simulator this is calculated by direct kinematic function. In reality it would be better to obtain $\boldsymbol{x}_e$ by measurements from sensors. A vision system could be used or, in indoor environments, a marker could be placed on the end-effector and tracked by a motion-capture system.

Figures 6.5 and 6.6 show the results of the circular trajectory tracing operation if, at $t = T_{\text{path}}/2$, the triangular impulse force described in 6.1 occurs. This force is considered as an unpredictable disturbance and is not considered when updating the momenta in (6.3). Two cases are distinguished:

- Figure 6.5: velocity-controlled manipulator without feedback ($\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_{\text{ref}}$)
- Figure 6.6: velocity-controlled manipulator with feedback ($\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_{\text{ref}} + \Delta\dot{\boldsymbol{q}}$). A gain K_P= 500 was chosen. As can be seen there's a great improvement in the tracking error.



(a) EE trajectory (cyan=desired, red=actual) and G_b trajectory (in blue)



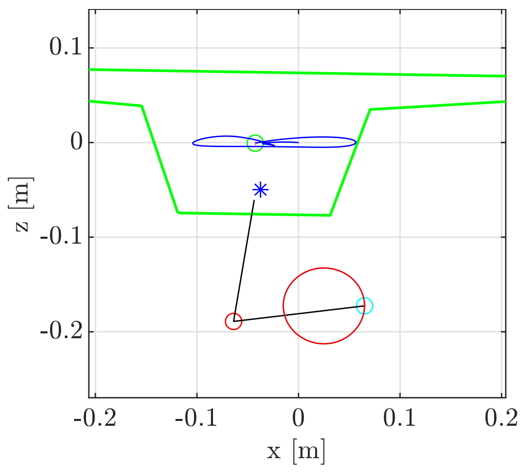
(b) norm of the EE position error over time $||\boldsymbol{e}(t)||$, red dashed line at $t = T_{\text{path}}/2$ (when impulse occurs), black vertical line at $t = T_{\text{path}}$

Figure 6.5: System response in case of impulse occurring at $t = T_{\text{path}}$. Manipulator controlled in velocity without feedback

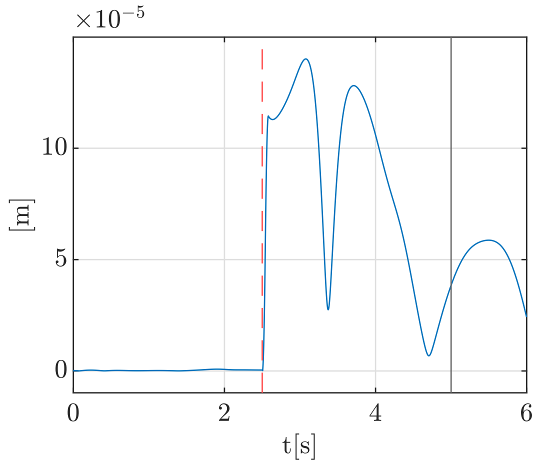
If one wants to use a feedback term that does not contemplate the inversion of the generalized Jacobian in real time, it is possible to take inspiration from the 3.31 (transposed Jacobian control) and choose:

$$\Delta\dot{\boldsymbol{q}} = [\boldsymbol{J}_{\text{gen}}]^T[\boldsymbol{K}_P]\boldsymbol{e} \tag{6.9}$$

This control is efficient and can be used also alone if the desired position is constant ($\dot{\boldsymbol{x}}_{e,\text{des}} = \boldsymbol{0}$). Figure 6.7 shows the result of a simulation in which the hand of the manipulator (controlled in velocity with $\Delta\dot{\boldsymbol{q}} = [\boldsymbol{J}_{\text{gen}}]^T[\boldsymbol{K}_P]\boldsymbol{e}$) is required to maintain the initial position despite the action of the triangular disturbance force at $t = T_{\text{path}}$. The gains of $[\boldsymbol{K}_P]$ matrix are $K_P = 5000$.

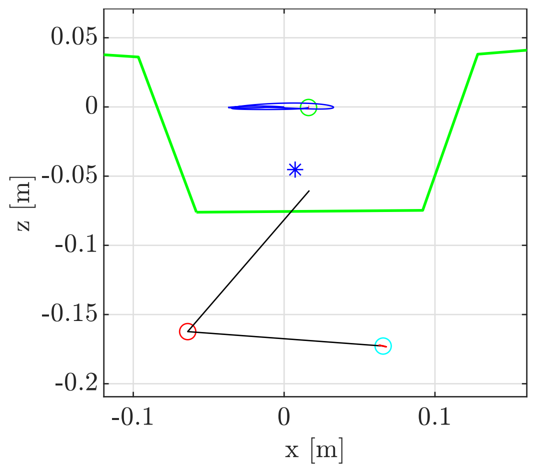


(a) EE trajectory (cyan=desired, red=actual) and G_b trajectory (in blue)

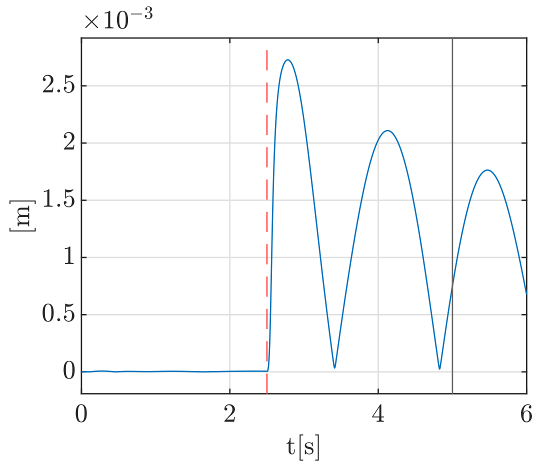


(b) norm of the EE position error over time $||\boldsymbol{e}(t)||$, red dashed line at $t = T_{\text{path}}/2$ (when impulse occurs), black vertical line at $t = T_{\text{path}}$

Figure 6.6: System response in case of impulse occurring at $t = T_{\text{path}}$ with Jacobian inverse control ($K_P = 500$)



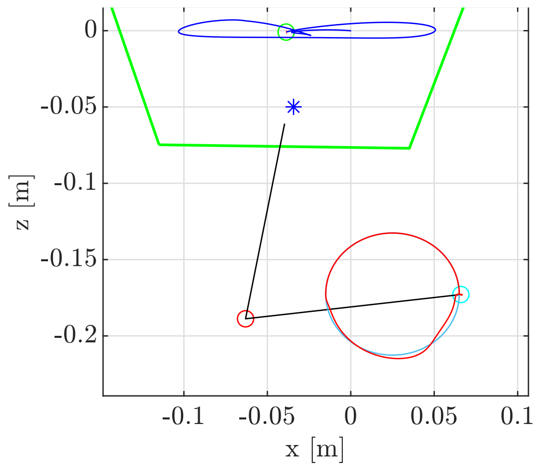
(a) In red EE actual trajectory and in blue G_b trajectory



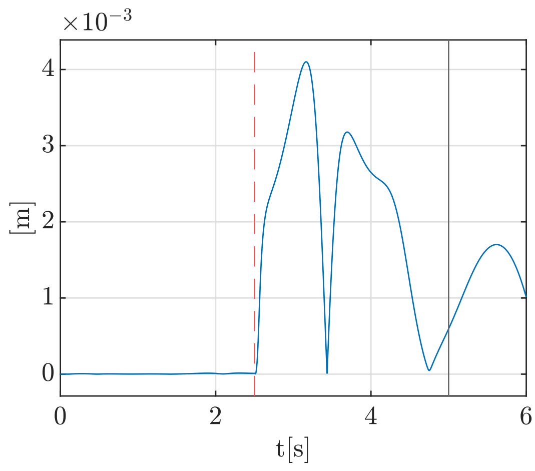
(b) norm of the EE position error over time $||\boldsymbol{e}(t)||$, red dashed line at $t = T_{\text{path}}/2$ (when impulse occurs), black vertical line at $t = T_{\text{path}}$

Figure 6.7: System response in case of impulse occurring at $t = T_{\text{path}}$ with Jacobian transpose control ($K_P = 5000$). Manipulator hand is required to stay fixed at initial position

In the case where a trajectory ($\dot{\boldsymbol{x}}_{e,\text{des}} \neq 0$) is required to be tracked and reference velocities can be precomputed through the (6.5) it is still possible to use $\Delta\dot{\boldsymbol{q}} = [\boldsymbol{J}_{\text{gen}}]^T[\boldsymbol{K}_P]\boldsymbol{e}$. The results, in the case of requiring the manipulator to follow the circular trajectory of 6.1, are shown in Figure 6.8. As can be seen, performances drops considerably compared to the case with inverse Jacobian feedback ($||\boldsymbol{e}(t)||$ of Figure 6.8b is two orders of magnitude greater than that in Figure 6.6b).



(a) EE trajectory (cyan=desired, red=actual) and G_b trajectory (in blue)



(b) norm of the EE position error over time $||\boldsymbol{e}(t)||$, red dashed line at $t = T_{\text{path}}/2$ (when impulse occurs), black vertical line at $t = T_{\text{path}}$

Figure 6.8: System response in case of impulse occurring at $t = T_{\text{path}}$ with Jacobian transpose control ($K_P = 5000$). Manipulator hand is required to track the circular trajectory detailed in 6.1

Better performance can be achieved by increasing gains, however, it should be noted that this also increases the danger of instability in real implementation where controllers are forced to process noisy signals. Transposed Jacobian control does not prove equal to inverse Jacobian control for trajectory tracking, but it has a number of advantages over the latter that make it preferable for less demanding cases. Calculating the transpose is much less onerous than calculating the inverse, making the online application faster. Also, in the case of kinematic singularity the transposed Jacobian algorithm does not fail unlike the inverse Jacobian algorithm.

6.3 Resolved Acceleration Control (RAC)

In this section, a second-order algorithm for controlling an aerial manipulator at the acceleration level is presented. It was taken as reference the algorithm developed in [21] for space manipulators.

Assume that the trajectory of the end-effector at the level of accelerations ($\ddot{\boldsymbol{x}}_{e,\text{des}}$) is given and that the generalized external forces ($\boldsymbol{F}_e = \dot{\boldsymbol{h}}$) acting on the system are known. Equation (4.25) that here is reported

$$\frac{d}{dt}\begin{Bmatrix}\boldsymbol{p}\\ \boldsymbol{K}\end{Bmatrix} = \dot{\boldsymbol{h}} = \boldsymbol{F}_e = [\boldsymbol{H}_b]\ddot{\boldsymbol{x}}_b + [\boldsymbol{H}_{bm}]\ddot{\boldsymbol{q}} + [\dot{\boldsymbol{H}}_b]\dot{\boldsymbol{x}}_b + [\dot{\boldsymbol{H}}_{bm}]\dot{\boldsymbol{q}} \tag{6.10}$$

can be rewritten as:

$$[\boldsymbol{H}_b]\ddot{\boldsymbol{x}}_b + [\boldsymbol{H}_{bm}]\ddot{\boldsymbol{q}} = \dot{\boldsymbol{h}} - \left([\dot{\boldsymbol{H}}_b]\dot{\boldsymbol{x}}_b + [\dot{\boldsymbol{H}}_{bm}]\dot{\boldsymbol{q}}\right) \doteq \boldsymbol{z_1} \tag{6.11}$$

In this form the equation represents an acceleration constraint. Along with (4.25), consider the kinematic equation (4.14)

$$\ddot{\boldsymbol{x}}_e = [\boldsymbol{J}_b]\ddot{\boldsymbol{x}}_b + [\dot{\boldsymbol{J}}_b]\dot{\boldsymbol{x}}_b + [\boldsymbol{J}_m]\ddot{\boldsymbol{q}} + [\dot{\boldsymbol{J}}_m]\dot{\boldsymbol{q}} \tag{6.12}$$

that can be rewritten as:

$$[\boldsymbol{J}_b]\ddot{\boldsymbol{x}}_b + [\boldsymbol{J}_m]\ddot{\boldsymbol{q}} = \ddot{\boldsymbol{x}}_{e,\text{des}} - \left([\dot{\boldsymbol{J}}_b]\dot{\boldsymbol{x}}_b + [\dot{\boldsymbol{J}}_m]\dot{\boldsymbol{q}}\right) \doteq \boldsymbol{z}_2 \tag{6.13}$$

Equations (6.11) and (6.13) can be summarized in the following single equation:

$$\begin{bmatrix}[\boldsymbol{H}_b] & [\boldsymbol{H}_{bm}]\\ [\boldsymbol{J}_b] & [\boldsymbol{J}_m]\end{bmatrix}\begin{Bmatrix}\ddot{\boldsymbol{x}}_b\\ \ddot{\boldsymbol{q}}\end{Bmatrix} = \begin{Bmatrix}\boldsymbol{z}_1\\ \boldsymbol{z}_2\end{Bmatrix} \tag{6.14}$$

For an UAM with a n DoFs manipulator, the coefficient matrix in Equation (6.14) is of dimensions $12 \times (n+6)$. In this work $n = 2$ so the matrix is 12×8. Equation (6.14) can be solved for accelerations yielding:

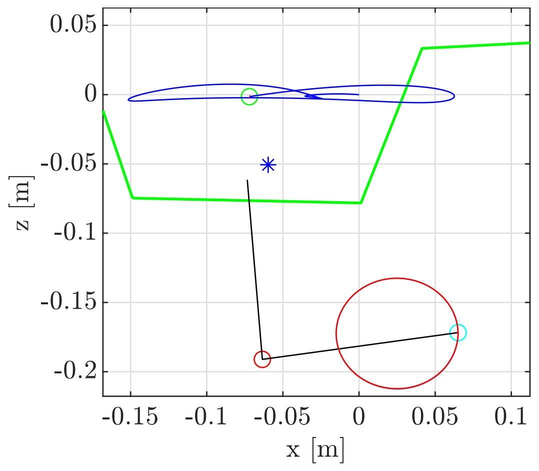
$$\begin{Bmatrix}\ddot{\boldsymbol{x}}_b\\ \ddot{\boldsymbol{q}}\end{Bmatrix} = \begin{bmatrix}[\boldsymbol{H}_b] & [\boldsymbol{H}_{bm}]\\ [\boldsymbol{J}_b] & [\boldsymbol{J}_m]\end{bmatrix}^{-1}\begin{Bmatrix}\boldsymbol{z}_1\\ \boldsymbol{z}_2\end{Bmatrix} \tag{6.15}$$

Equation (6.15) is the key equation of RAC algorithm that allows to solve the inverse kinematics of an UAM at accelerations level. Joint velocities $\dot{\boldsymbol{q}}$ inside $\boldsymbol{z_2}$

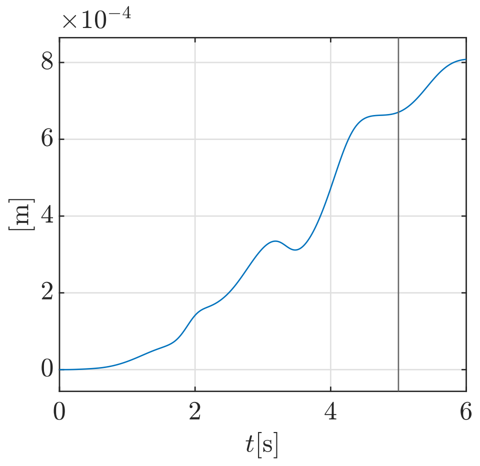
can be measured with the help of sensors. The linear and angular velocities of the base can also be measured, or they can be computed from the momentum constraint (4.24) as:

$$\dot{\boldsymbol{x}}_b = [\boldsymbol{H}_b]^{-1} (\boldsymbol{h} - [\boldsymbol{H}_{bm}]\dot{\boldsymbol{q}}) \tag{6.16}$$

As for RMRC algorithm, RAC algorithm can work effectively without feedback if all the forces and torques acting on the system can be estimated sufficiently accurately so that the term $\dot{\boldsymbol{h}}$ in $\boldsymbol{z}_1$ is updated correctly in the controller. Figure 6.9 shows the results of inverse kinematics when all forces acting on the system are known. As for RMRC experiments, the only forces that are acting on the UAM in this case are the weight force and the controls U_1 and U_2.



(a) EE trajectory (cyan=desired, red=actual) and G_b trajectory (in blue)



(b) norm of the EE position error over time $||\boldsymbol{e}(t)||$, black vertical line at $t = T_{\text{path}}$

Figure 6.9: Circular trajectory tracking through RAC. Acceleration profiles of manipulator motors are determined by Eq.(6.15). The only forces acting on the system are the weight force, U_1 and U_2, which are known and updated in the controller

The main difficulty of resolved acceleration control of flying-base manipulator as compared to ground-fixed manipulator is due to the fact that, in resolving acceleration, one has to consider the differentiated momentum constraint (6.11), together with the kinematic constraint (6.13). For these manipulators, kinematics and dynamics cannot be treated separately, which greatly complicates control.

Compared with RMRC, RAC has the main advantage of being able to operate in the forces/torques domain, which can be estimated directly if the UAM has the right sensory configuration. In the RMRC algorithm, forces must be integrated to obtain linear and angular momenta estimates, and this can cause a lot of error buildup.

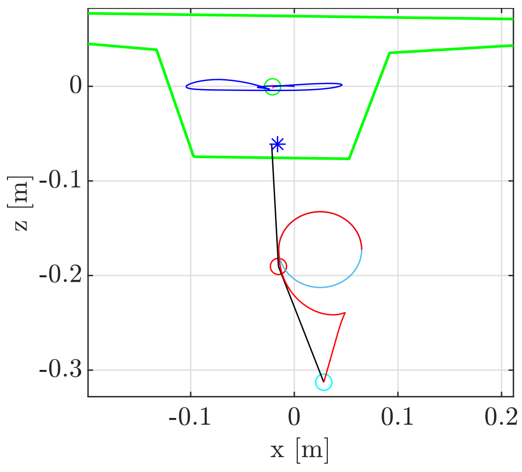
Also in the case of RAC it may be convenient to add a feedback term, resulting in a second-order CLIK algorithm. This could serve to counteract possible errors due to inaccuracies in the knowledge of UAM parameters and/or acting forces. To add an operational-space error based feedback, as with RMRC, $\boldsymbol{z}_2$ is modified as:

$$\boldsymbol{z}_2 \doteq \ddot{\boldsymbol{x}}_{e,\mathrm{des}} - \left([\dot{\boldsymbol{J}}_b]\dot{\boldsymbol{x}}_b + [\dot{\boldsymbol{J}}_m]\dot{\boldsymbol{q}}\right) + ([\boldsymbol{K}_P]\boldsymbol{e} + [\boldsymbol{K}_D]\dot{\boldsymbol{e}}) \tag{6.17}$$

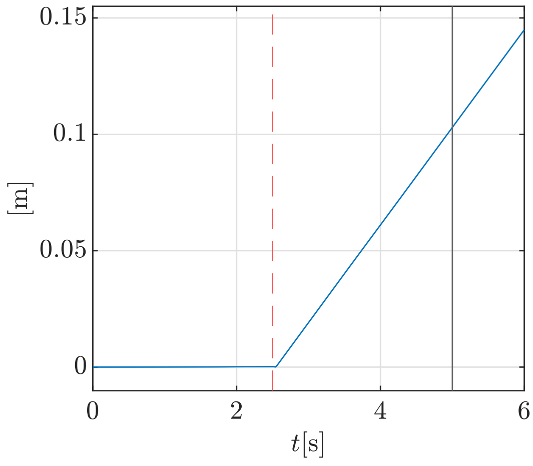
where $[\boldsymbol{K}_P]$ and $[\boldsymbol{K}_D]$ are diagonal positive definite matrices of gains and $\dot{\boldsymbol{e}}$ is the time derivative of the end-effector position error (in the two-dimensional case it coincides with the error on the end-effector linear velocity). Also in this case the acceleration input given to the motors can be decomposed as:

$$\ddot{\boldsymbol{q}} = \ddot{\boldsymbol{q}}_{\mathrm{ref,ik}} + \Delta\ddot{\boldsymbol{q}}_{\mathrm{feedback}} \tag{6.18}$$

In the simulator accelerations $\ddot{\boldsymbol{q}}_{\mathrm{ref,ik}}$ can be be calculated both a priori or online during the maneuver. Figures 6.10 and 6.11 show the results of two simulations in which the UAM is controlled by RAC algorithm and at $t = T_{\mathrm{path}}/2$ is disturbed by the triangular impulse force of 6.1. Figure 6.10 shows the result on the system with feedback inactive while in Figure 6.11 the feedback is active with gains $K_P = 500$ and $K_D = 100$.

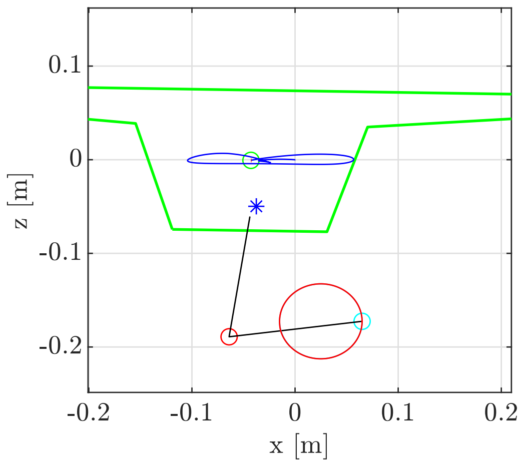


(a) EE trajectory (cyan=desired, red=actual) and G_b trajectory (in blue)

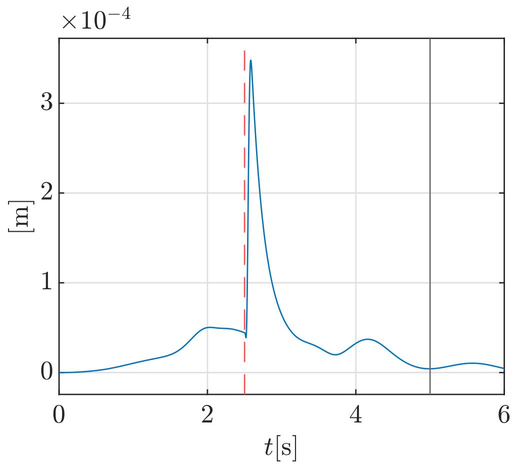


(b) norm of the EE position error over time $||\boldsymbol{e}(t)||$, black vertical line at $t = T_{\text{path}}$

Figure 6.10: System response in case of impulse occurring at time $t = T_{\text{path}}/2$. Manipulator controlled through RAC without feedback ($K_P = 0,\ K_D = 0$)



(a) EE trajectory (cyan=desired, red=actual) and G_b trajectory (in blue)



(b) norm of the EE position error over time $||\boldsymbol{e}(t)||$, black vertical line at $t = T_{\text{path}}$

Figure 6.11: System response in case of impulse occurring at time $t = T_{\text{path}}/2$. Manipulator controlled through RAC with feedback ($K_P = 500,\ K_D = 100$)

Computational details of RAC

This subsection summarized the steps of the computation required for solving the inverse kinematics of UAM systems at acceleration level at each timestep.

1. Measure $\dot{\boldsymbol{x}}_b$, $\dot{\boldsymbol{q}}$ and $\boldsymbol{x}_b$, $\boldsymbol{q}$. In the simulator these velocities and positions come from the single and double numerical integration of the accelerations calculated through forward dynamics procedure described in 4.3.4.

2. Compute the rotation matrices $[\boldsymbol{R}_{iI}]$ for transforming coordinates known in the i-th frame $\mathcal{F}_i$ in coordinates in the inertial frame $\mathcal{F}_I$.

3. Compute the positions of the links centers of mass $\boldsymbol{r}_{G_i}$.

4. Compute $\left([\dot{\boldsymbol{J}}_b]\dot{\boldsymbol{x}}_b + [\dot{\boldsymbol{J}}_m]\dot{\boldsymbol{q}}\right)$. For doing this observe that from 6.13 it is possible to deduce that:

$$[\dot{\boldsymbol{J}}_b]\dot{\boldsymbol{x}}_b + [\dot{\boldsymbol{J}}_m]\dot{\boldsymbol{q}} = \ddot{\boldsymbol{x}}_e\big|_{\ddot{\boldsymbol{x}}_b=\ddot{\boldsymbol{q}}=\boldsymbol{0}} = \ddot{\boldsymbol{x}}_{e,0} \tag{6.19}$$

This acceleration is calculated through the forward recursion of Newton-Euler recursive scheme (see 4.2.1) under the constraint $\ddot{\boldsymbol{x}}_b = \ddot{\boldsymbol{q}} = \boldsymbol{0}$

5. Compute $\left([\dot{\boldsymbol{H}}_b]\dot{\boldsymbol{x}}_b + [\dot{\boldsymbol{H}}_{bm}]\dot{\boldsymbol{q}}\right)$. From 6.11 it is possible to deduce the relation:

$$[\dot{\boldsymbol{H}}_b]\dot{\boldsymbol{x}}_b + [\dot{\boldsymbol{H}}_{bm}]\dot{\boldsymbol{q}} = \dot{\boldsymbol{h}}\bigg|_{\ddot{\boldsymbol{x}}_b=\ddot{\boldsymbol{q}}=\boldsymbol{0}} = \dot{\boldsymbol{h}}_0 \tag{6.20}$$

$\dot{\boldsymbol{h}}_0$ represents the generalized inertial force acting on the system when $\ddot{\boldsymbol{x}}_b = \ddot{\boldsymbol{q}} = \boldsymbol{0}$. It is calculated by the backward recursion of the Newton-Euler scheme applied in the previous step.

6. Compute $[\boldsymbol{H}_b]^{-1}$ (see 4.18). This can be computed using the formula of the inverse of a block matrix [14] as:

$$[\boldsymbol{H}_b]^{-1} = \begin{bmatrix} [\mathbb{I}_3] & -[\boldsymbol{S}(\boldsymbol{r}_{G_b,G_i})]^T \\ [\boldsymbol{0}] & [\mathbb{I}_3] \end{bmatrix} \begin{bmatrix} (1/m_{\text{tot}})[\mathbb{I}_3] & [\boldsymbol{0}] \\ [\boldsymbol{0}] & [\boldsymbol{s}]^{-1} \end{bmatrix} \begin{bmatrix} [\mathbb{I}_3] & [\boldsymbol{0}] \\ -[\boldsymbol{S}(\boldsymbol{r}_{G_b,G_i})] & [\mathbb{I}_3] \end{bmatrix} \tag{6.21}$$

where $[\boldsymbol{s}] = [\boldsymbol{H}_\omega] + m_{\text{tot}}[\boldsymbol{S}(\boldsymbol{r}_{Gb,G})]^2$.

7. Compute $[\boldsymbol{H}_{bm}]$ from Equation (4.20)

8. Compute $\ddot{\boldsymbol{x}}_b$ and $\ddot{\boldsymbol{q}}$ through Equation (6.15). The inverse of the coefficient matrix can be computed with the same formula mentioned in step 6 obtaining:

$$\begin{bmatrix} [\boldsymbol{H}_b] & [\boldsymbol{H}_{bm}] \\ [\boldsymbol{J}_b] & [\boldsymbol{J}_m] \end{bmatrix}^{-1} =$$

$$= \begin{bmatrix} [\mathbb{I}_6] & -[\boldsymbol{H}_b]^{-1}[\boldsymbol{H}_{bm}] \\ [\mathbf{0}] & [\mathbb{I}_n] \end{bmatrix} \begin{bmatrix} [\boldsymbol{H}_b]^{-1} & [\mathbf{0}] \\ [\mathbf{0}] & [\boldsymbol{J}_{\text{gen}}]^{-1} \end{bmatrix} \begin{bmatrix} \mathbb{I}_6 & \mathbf{0} \\ -[\boldsymbol{J}_b][\boldsymbol{H}_b]^{-1} & [\mathbb{I}_n] \end{bmatrix} \tag{6.22}$$

Instead of computing the inverse by multiplicating the three matrices, it is computationally less expensive to repeat three times the matrix vector product that comes out by substituting (6.22) into (6.15). Note that if the manipulator passes through a singular configuration, since $[\boldsymbol{J}_{\text{gen}}]$ is not invertible, the coefficient matrix is not invertible thus the algorithm fails.

Chapter 7

RAC tests

This chapter reports the results of several virtual experiments on an UAM controlled at acceleration level through the RAC algorithm presented in Chapter 6. Experiments are all performed in the MATLAB simulator presented in 4. The experiments objective is the same of that described in 6.1: making the end-effector to follow a circular trajectory of diameter D in a time T_{path} and stay on the final point for a time T_{post}. The parameters of the UAM protagonist of the simulations are the same reported in Table 6.1.

The chapter objective is to present the performances of the RAC algorithm when the system is subjected to the action of more realistic disturbances than the impulse shown in Chapter 6. The modeling of such disturbances was discussed in Chapter 5. For each of these, the response of the system with feedback control off (gains equal to zero) will be presented first and then the response when feedback is on. In the case of disturbances of a random nature, multiple experiments will be performed and then an average of the results will be reported. As measure for evaluating the performance of the control system, the trend of the end-effector position error norm over time $||\boldsymbol{e}(t)||$ was chosen.

At the end are reported the results of a sensitivity analysis done for a preliminary tuning of the feedback term gains and a comparison between RAC and RMRC.

7.1 Response to wind disturbance

UAMs will be requested to operate in both outdoor and indoor scenarios. In both cases the system may be disturbed by aerodynamic forces (i.e. wind in outdoor applications) causing significant movements of the manipulator base. RAC algorithm must prove that it can make the manipulator complete the task equally, despite these unpredictable movements.

First the system is tested assuming that the disturbance is a wind gust modelled as described in 5.1.1. Aerodynamic forces on the UAV caused by this wind are plotted in Figure 7.1. Figure 7.2 shows the response of the system with feedback loop deactivated, while in Figure 7.3 is shown the response when the feedback is active with $K_P = 500$ and $K_D = 100$.

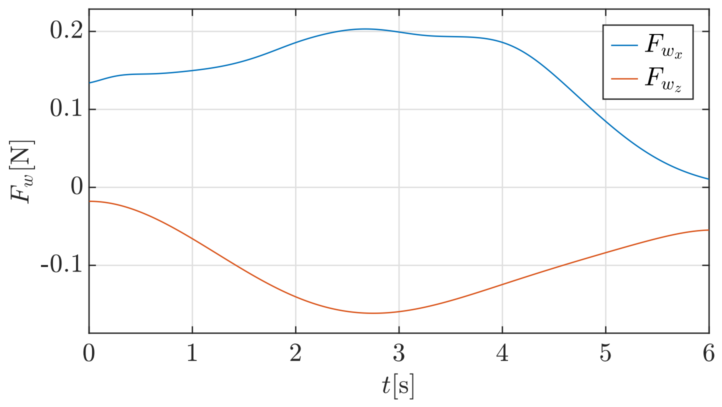
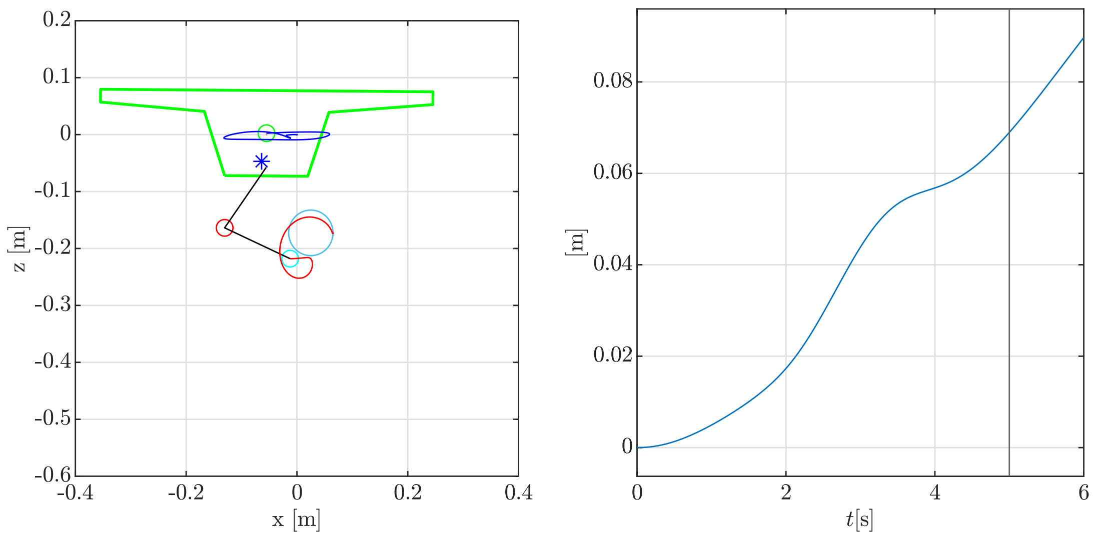


Figure 7.1: Aerodynamic forces on the UAV for the experiments of Figure 7.2 and 7.3. $n = 40$ and $\Omega_i \in [0.1, 1.5]$ rad/s

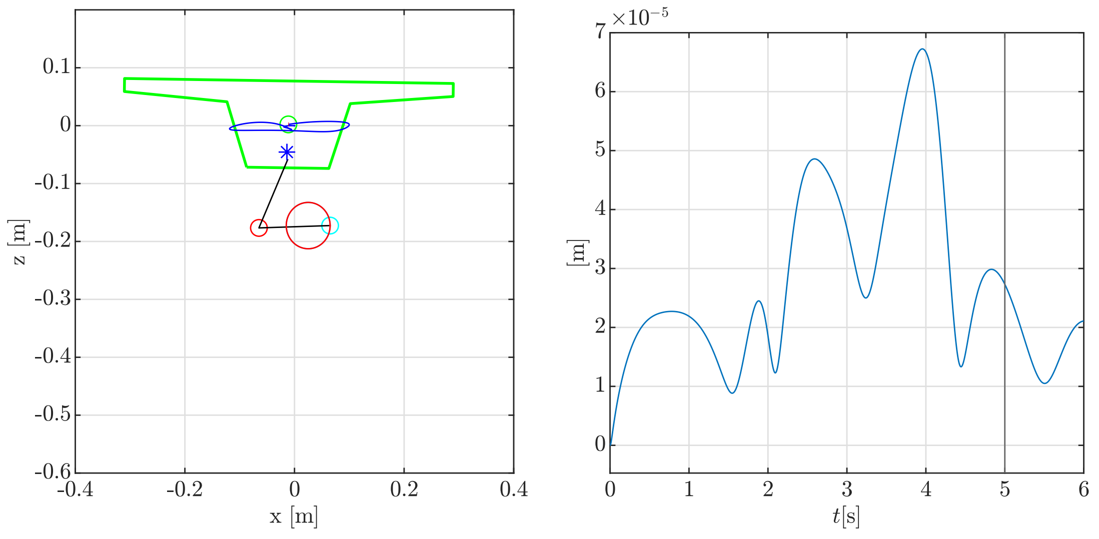
Since the disturbance is random in nature, to give a more accurate picture of the control system's ability to reject it, a multiplicity of experiments were performed. Figure 7.4 shows the plot of the mean of the position error norm over time based on the results of 10 experiments. The gains were always $K_P = 500$ and $K_D = 100$.



(a) EE trajectory (cyan=desired, red=actual) and G_b trajectory (in blue)

(b) norm of the EE position error over time $||\boldsymbol{e}(t)||$, black vertical line at $t = T_{\text{path}}$

Figure 7.2: System response to wind gust when controlled trough RAC with no feedback ($K_P, K_D = 0$)



(a) EE trajectory (cyan=desired, red=actual) and G_b trajectory (in blue)

(b) norm of the EE position error over time $||\boldsymbol{e}(t)||$, black vertical line at $t = T_{\text{path}}$

Figure 7.3: System response to wind gust when controlled through RAC with feedback ($K_P = 500, K_D = 100$)

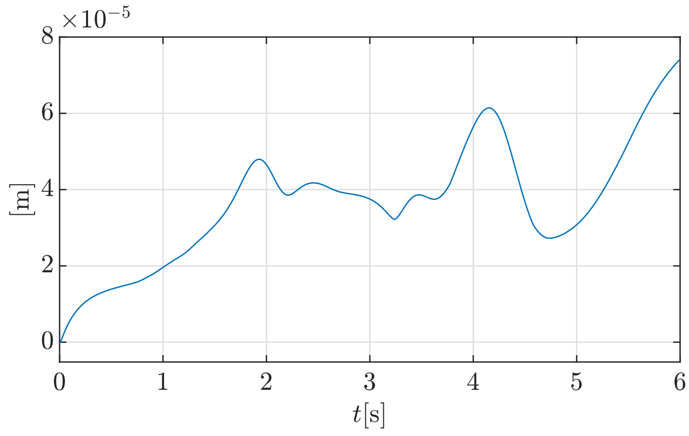


Figure 7.4: Mean value of $||\boldsymbol{e}(t)||$ over 10 virtual experiments. UAM subjected to wind force and controlled through closed-loop RAC algorithm ($K_P = 500, K_D = 100$)

From what can be seen in Figures 7.2, 7.3 and 7.4 it is possible to affirm that RAC algorithm shows very good performances when the UAM is subjected to external unknown forces. The norm of the end-effector position error stayed under the tenth of millimeter in simulations.

In indoor applications, UAMs may not experience wind gusts but still experience the effect of other disturbances of aerodynamic nature that are difficult to quantify. Normally these disturbances are almost random in nature, and have an higher frequencies than the wind experienced outdoors. As was mentioned in Chapter 5, the implemented wind model can be modified in its frequency content. This can be done, for example, by choosing higher frequencies for the sine waves that make up the wind speeds. In this way, it is possible to quickly obtain a very simple model, that maybe is not accurate in magnitude, but can simulate to some degree the aerodynamic effects experienced in indoor scenarios. Figure 7.5 shows an example of aerodynamic force profile when the frequencies of the sinusoids in Dryden model are picked from the interval $[1, 30]$ rad/s.

Similarly to what was done before, in Figure 7.6 is reported the plot of the mean value of $||\boldsymbol{e}(t)||$ over 10 experiments when the frequencies of Dryden model sinusoids are taken in the interval [1,30] rad/s (always $K_P = 500, K_D = 100$). Also in this case the RAC algorithm proves to be robust.

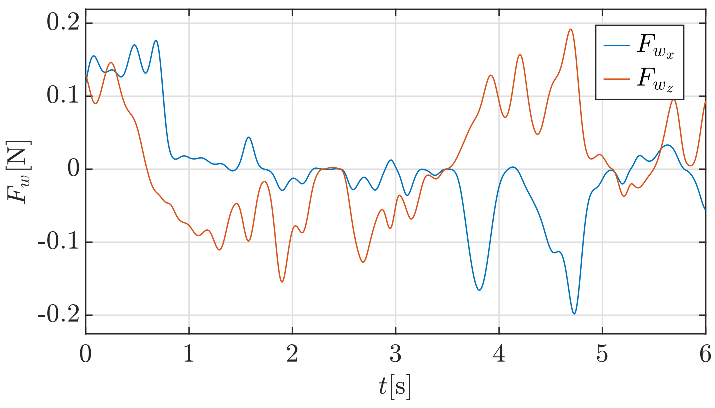


Figure 7.5: Example of aerodynamic forces profiles when frequencies of Dryden model sinusoids $\Omega_i \in [1, 30]$ rad/s

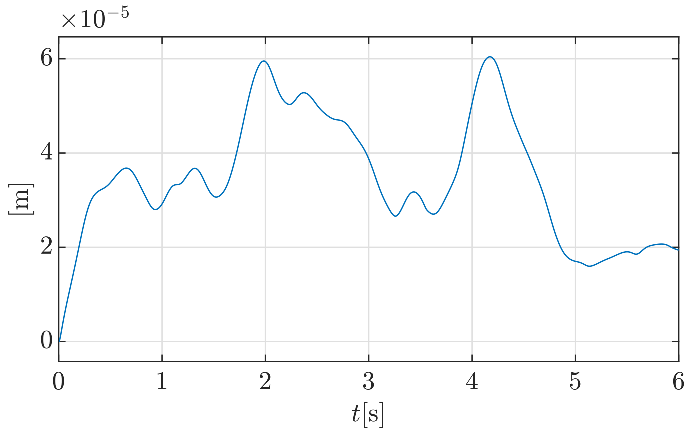
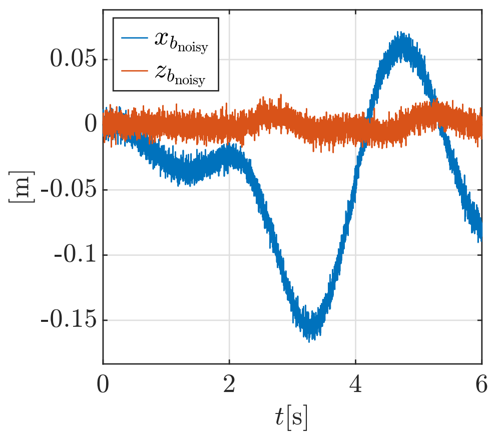
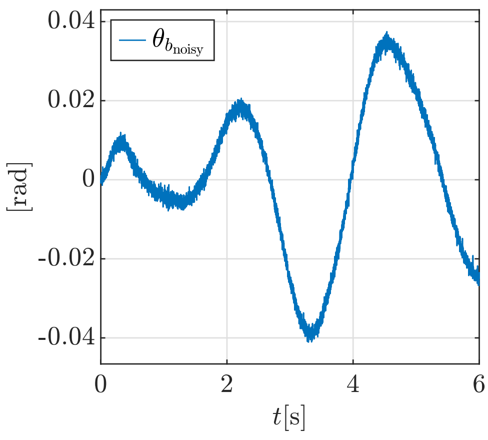


Figure 7.6: Mean value of $||\boldsymbol{e}(t)||$ over 10 virtual experiments with higher frequency disturbance. UAM controlled through closed-loop RAC algorithm$(K_P = 500, K_D = 100)$

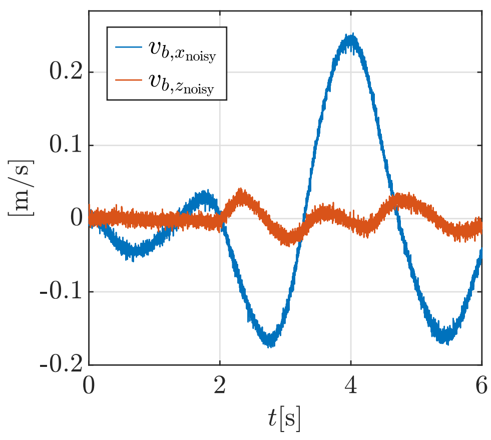
7.2 Response in case of noise in the measurement chain



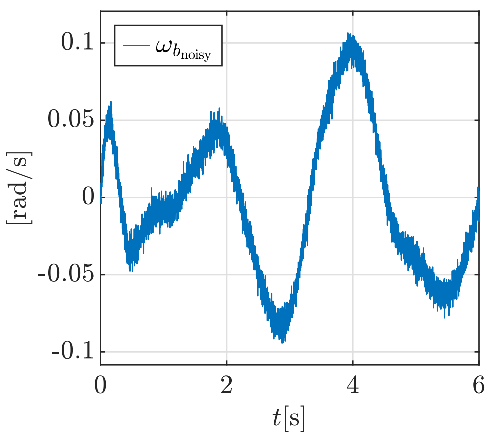
(a) UAV translations



(b) UAV rotations



(c) UAV linear velocities



(d) UAV angular velocity

Figure 7.7: UAV kinematic quantities corrupted by noise. UAM controlled through inverse kinematics algorithm at acceleration level

As discussed in chapter 5, it is inevitable that signals from the on-board sensors will be noisy. This can pose a serious threat to the proper operation and stability of the UAM control system since the feedback loop no longer relies on correct information about the system status. Therefore, it is important to verify that the RAC algorithm is robust to noise.

To do this, in the simulator are performed some experiments in which the signals of UAV kinematic quantities are corrupted with distributed Gaussian noise. The generic signal $x(t)$ thus become $\mathbf{X}(t) = x(t) + \sigma\mathbf{Z}$ with $\mathbf{Z} \sim \mathcal{N}(0,1)$ (see 5.2 for more details and Table 5.2 for std. deviations values).

Figure 7.7 shows the plots of disturbed signals (system controlled by RAC algorithm with no feedback). From Figure 7.8 it is possible to see how, even in the absence of unknown forces acting on the system, the presence of the noise leads the end-effector not to follow the desired trajectory correctly. In fact, noise affects both the calculation of the drone control forces U_1 and U_2 from the PID system and the online calculation of the terms $\dot{\boldsymbol{h}}_0$ and $\ddot{\boldsymbol{x}}_{e,0}$ in the RAC algorithm.

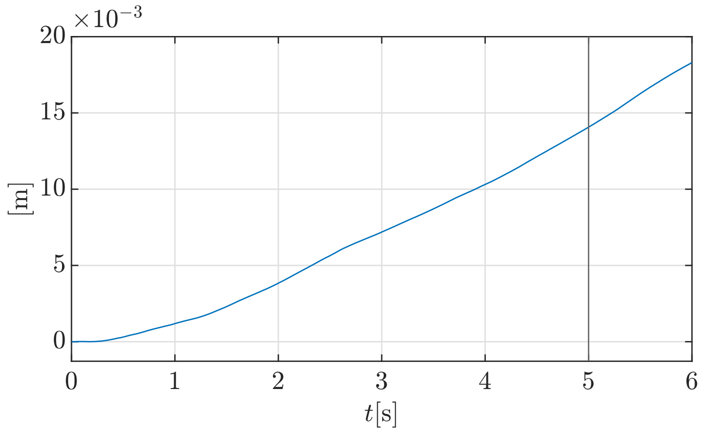


Figure 7.8: $||\boldsymbol{e}(t)||$ when UAM is controlled only through inverse kinematics algorithm at acceleration level $(K_P = 0, K_D = 0)$ in the presence of noise

To effectively test the RAC algorithm, it was assumed in the simulations that the impulsive disturbance described in 6.1 would act in the middle of the maneuver. In this way it is possible to see how the algorithm reacts to uncertainty about the actual position and velocity of the end-effector. Figures 7.9 and 7.10 compare the plots of $||\boldsymbol{e}(t)||$ in the case of inactive and active feedback.

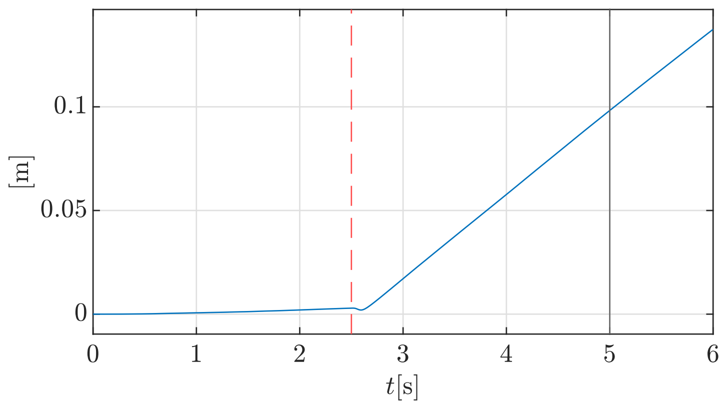


Figure 7.9: $||\boldsymbol{e}(t)||$ when a triangular impulsive force of maximum 10 N occurs at $t = T_{\text{path}}/2$ (vertical red dashed line). UAM is controlled only through inverse kinematics algorithm at acceleration level ($K_P = 0, K_D = 0$).

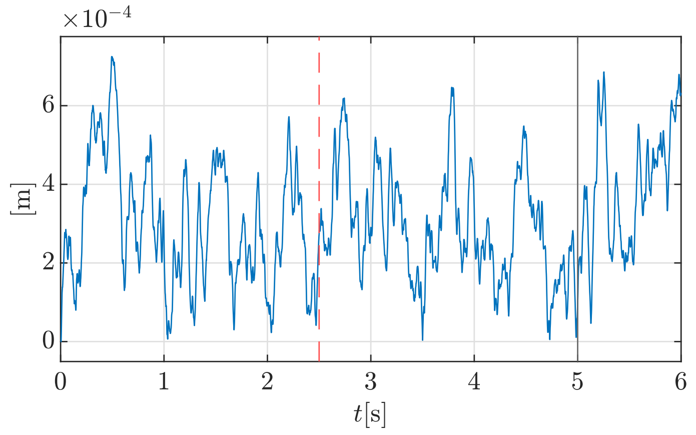


Figure 7.10: $||\boldsymbol{e}(t)||$ when a triangular impulsive force of maximum 10 N occurs at $t = T_{\text{path}}/2$ (vertical red dashed line). UAM is controlled with closed-loop RAC algorithm ($K_P = 500, K_D = 100$)

As can be seen from Figure 7.10, the RAC algorithm proves to be robust even in the presence of noise in the measurement chain. The error is greatly reduced with respect to the case of Figure 7.9 and remains bounded both before and after the impulse occurring.

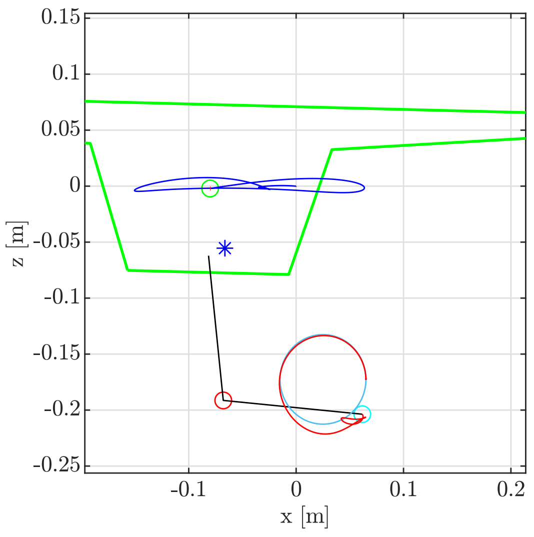
According to the simulation, the trajectory is irregular, and this may cast doubt on the effectiveness of the control. Actually, it must be said that unmanned systems such as UAVs nowadays implement sophisticated algorithms, such as the

Kalman filters, already mentioned in Chapter 2, to process data from sensors. Thanks to these both noise and uncertainty on the data are greatly reduced, making the scenario assumed in these simulations almost extreme. However, it should be emphasized that even in these scenarios RAC has shown that it can provide good performances while keeping the system stable.

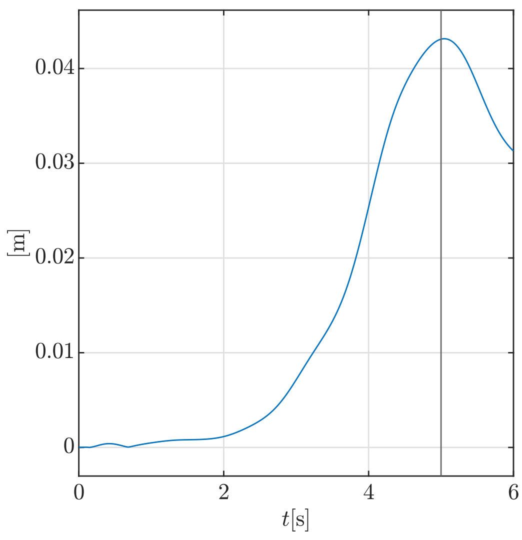
7.3 Robustness to parameter uncertainty test

In this section are reported the results of virtual experiments on the UAM in the case of uncertainty in the model parameters used by the control algorithm. In this way, the robustness of the system to parameter variation can be evaluated. The uncertainty is modelled as described in 5.3. Three categories are therefore distinguished and the response of the system in each of these cases is shown.

7.3.1 Control in the case of uncertain inertial parameters



(a) EE trajectory (cyan=desired, red=actual) and G_b trajectory (in blue)



(b) norm of the EE position error over time $||\boldsymbol{e}(t)||$, black vertical line at $t = T_{\text{path}}$

Figure 7.11: System response when controlled by open-loop inverse kinematic algorithm $(K_P, K_D = 0)$ in the case of model incorrect inertial parameters values.

In this case there is an uncertainty both on UAV inertial parameters (m_{UAV} and I_{G_b}) and on links (m_i and I_{G_i}). Figure 7.11 shows the response of the system when accelerations are computed only by mean of inverse kinematic algorithm. Since also in this case the nature of the disturbance is random, several experiments were performed. Figure 7.12 show the plot of the mean value of $||\boldsymbol{e}(t)||$ for 5 experiments.

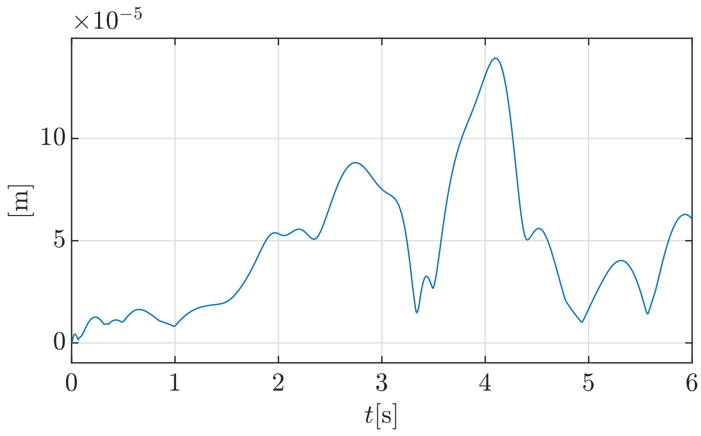
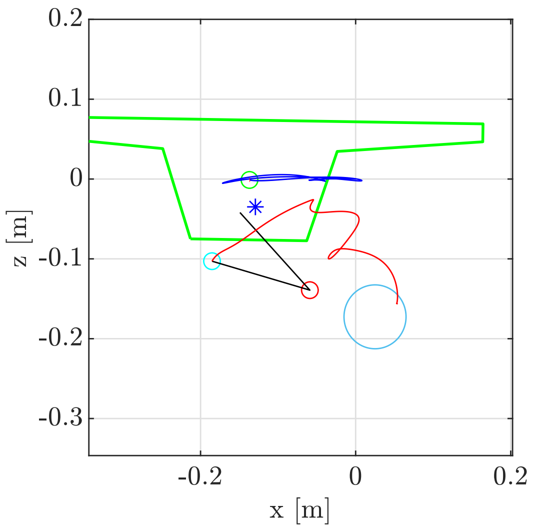


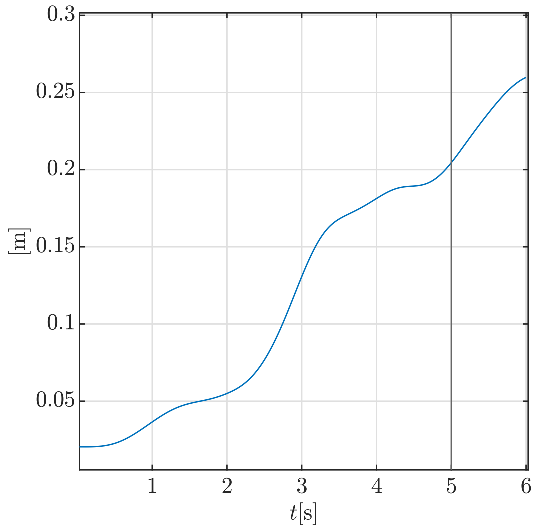
Figure 7.12: Mean value of $||\boldsymbol{e}(t)||$ over 5 virtual experiments. Uncertain inertial parameters. System controlled by closed-loop RAC algorithm ($K_P = 500, K_D = 100$)

7.3.2 Control in the case of geometric uncertainties

Also errors in the geometric model of the UAM can lead to trajectory tracking errors. Figure 7.13 shows the tracking error in case of incorrect knowledge of the centers of mass location of the links and the UAV (modelled as described in 5.3.2). CoG locations are random for every simulations as explained in 5.3.2. In Figure 7.14 is reported the plot of the average error obtained in 5 experiments. Since the position error of the first joint inevitably causes an initial positioning error of the end-effector, the first second of the plot is omitted. As can be seen from the figure, when the feedback loop is active, the system takes little time to bring itself back into position.



(a) EE trajectory (cyan=desired, red=actual) and G_b trajectory (in blue)



(b) norm of the EE position error over time $||\boldsymbol{e}(t)||$, black vertical line at $t = T_{\text{path}}$

Figure 7.13: System response in the case of model incorrect centers of mass location informations $(K_P, K_D = 0)$

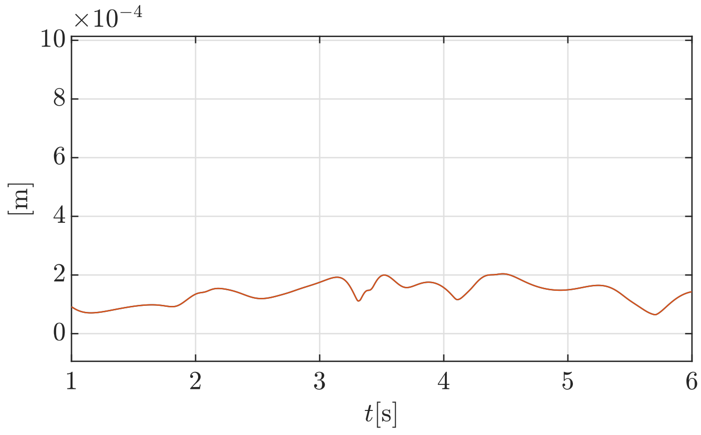
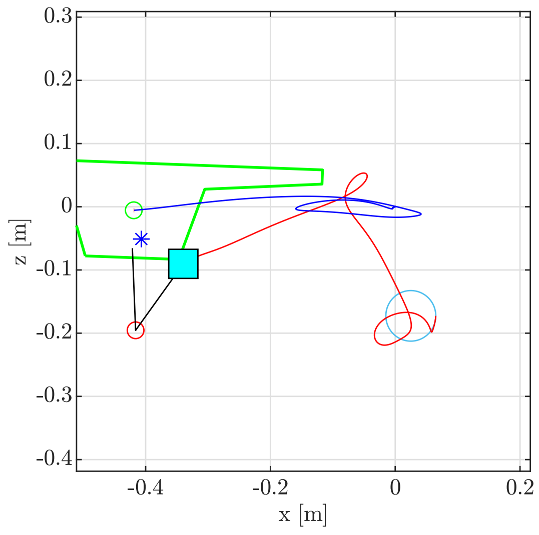


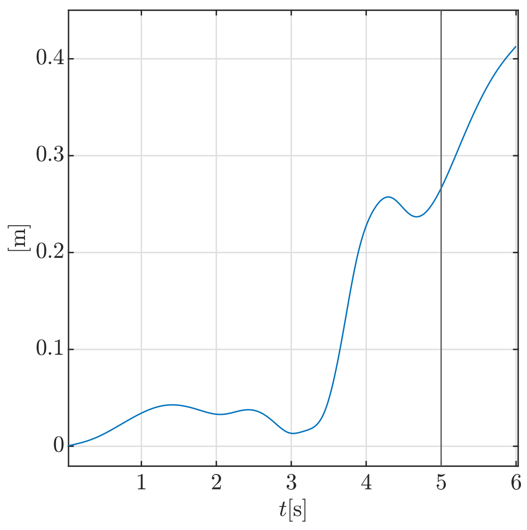
Figure 7.14: Mean value of $||\boldsymbol{e}(t)||$ over 5 virtual experiments. Uncertain CoG locations. System controlled in acceleration with feedback active $(K_P = 500, K_D = 100)$

7.3.3 Control in the case of unknown mass of the carried load

In this case it is assumed that the end organ of the manipulator carries a load of completely unknown mass (equal to 0.5 kg in the simulations).

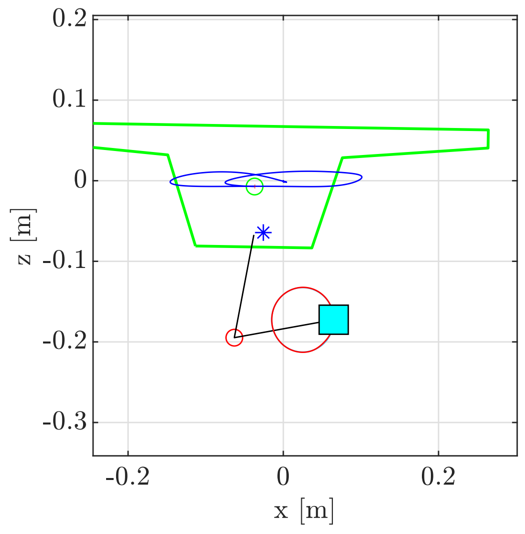


(a) EE trajectory (cyan=desired, red=actual, square=load mass) and G_b trajectory (in blue)

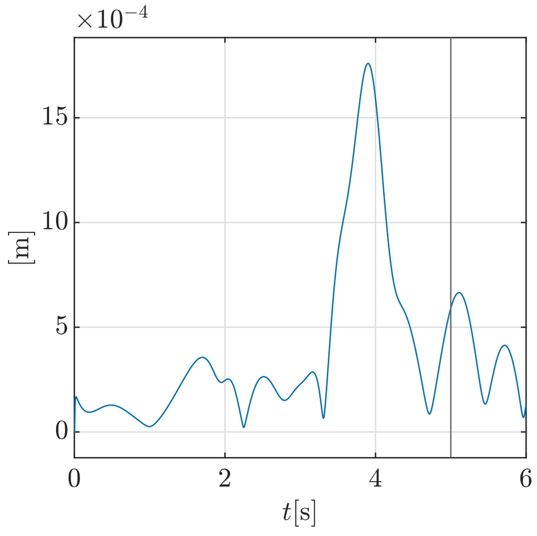


(b) norm of the EE position error over time $||\boldsymbol{e}(t)||$, black vertical line at $t = T_{\text{path}}$

Figure 7.15: System response in the case of unknown carried mass by the EE ($K_P, K_D = 0$, $m_{\text{load}} = 0.5$kg)



(a) EE trajectory (cyan=desired, red=actual, square=load mass) and G_b trajectory (in blue)



(b) norm of the EE position error over time $||\boldsymbol{e}(t)||$, black vertical line at $t = T_{\text{path}}$

Figure 7.16: System response in the case of unknown carried mass by the EE ($K_P = 500, K_D = 100$, $m_{\text{load}} = 0.5$kg)

The transport of the load is modelled as described in 5.3.3. Figure 7.15 shows the system response when the feedback loop is deactivated. As can be seen the actual trajectory gets very far from the desired one. Figure 7.16 shows how adding the feedback makes the system accomplishing the trajectory with a low error.

7.4 Response in the case of multiple disturbances

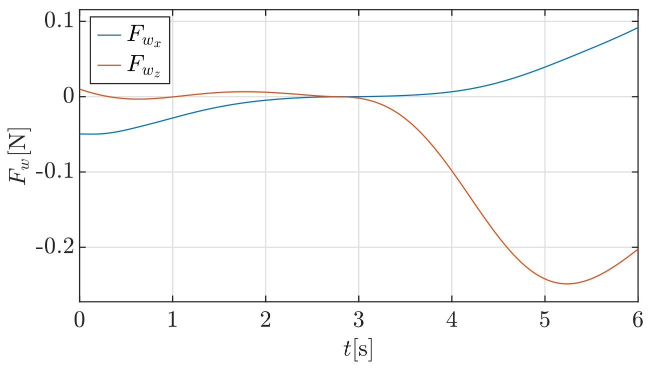


Figure 7.17: Aerodynamic forces on the UAV for the experiment of Figure 7.18 . $n = 30$ and $\Omega_i \in [0.1, 1.5]$ rad/s

In this section the algorithm is tested assuming that on the the UAM is acting an aerodynamic disturbance (force profile plotted like in Figure 7.17) and that the the signal coming from sensors are noisy. In the middle of the simulation the triangular impulse of 6.1 occurs. The result in term of norm of the tracking error is plotted in Figure 7.18. Also in this case RAC shows good performances with the distance of the end-effector from desired trajectory staying below the millimeter.

7.5 Sensitivity analysis for preliminary gains tuning

This section reports the results of the sensitivity analysis done by varying the control gains. The goal is to find the most appropriate combination of K_P and K_D for UAM control without the danger of the system becoming unstable. In

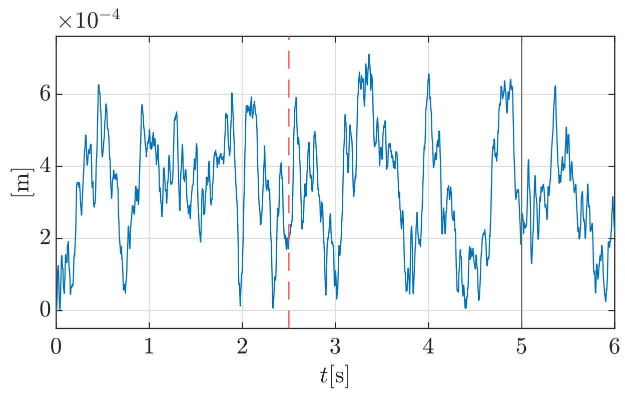


Figure 7.18: Response of the system controlled by closed-loop RAC algorithm ($K_P = 500, K_D = 100$) in the case multiple disturbances. At $t = T_{\text{path}}/2$ the impulse described in 6.1 occurs

the simulations performed, the UAM was subject to the action of aerodynamic disturbance, noise, and impulsive force at $t = T_{\text{path}}/2$. The combinations of the gains were created by taking the gains from the following vectors:

- K_P from $\{10, 100, 500, 1000, 5000\}$
- K_D from $\{0.1, 1, 10, 100, 1000\}$

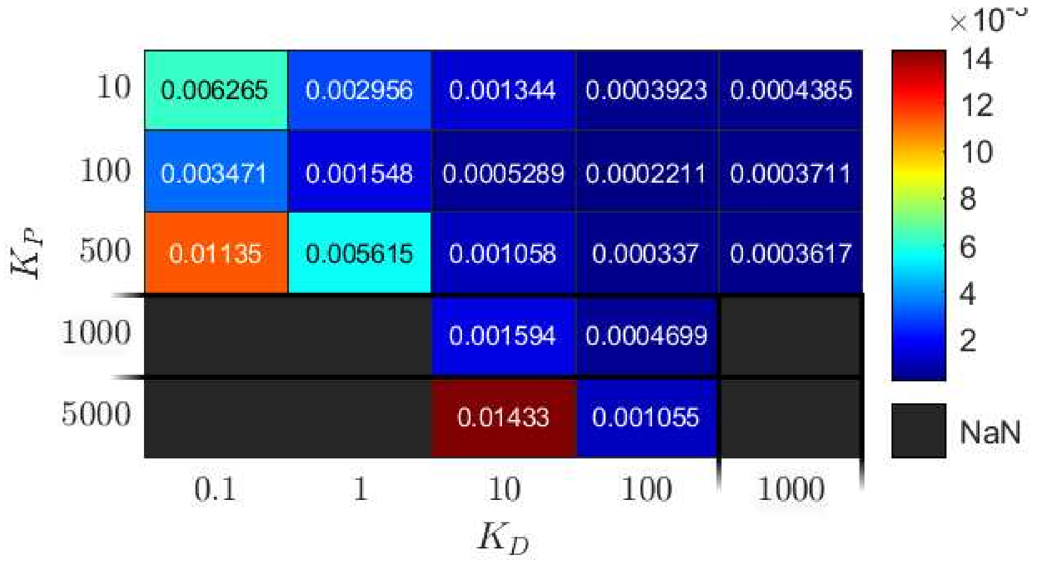


Figure 7.19: Results of the sensitivity analysis. Numbers inside the squares represent the final mean value "mean($||\boldsymbol{e}(t)||$)". Black squares indicate that the corresponding combination of K_P,K_D led to instability

This yields 25 combinations of K_P,K_D. Given the random nature of the aerodynamic disturbance, 3 experiments were performed for each combination. The profile of the average trend of the position error norm for the three experiments was then calculated and from this the average value of the error "mean($||\boldsymbol{e}(t)||$)". This benchmark value is represented by the color of the squares in the heatmap of Figure 7.19.

As the heatmap shows, the combination used in this chapter is one of the best possible. Results suggest to take gains K_P, K_D in the hundreds.

7.6 Comparison between RAC and RMRC algorithms

To conclude, a comparison made between the RMRC and RAC algorithms is reported. The comparison is made by requiring the UAM to follow the usual circular trajectory, first commanded in velocity, then in acceleration, with the UAV under the effect of the same aerodynamic force and impulsive force at $t = T_{\text{path}}/2$. To make the comparison results more readable, noise was ignored in the two experiments.

Figure 7.20 shows the plots of the force acting in the experiments.

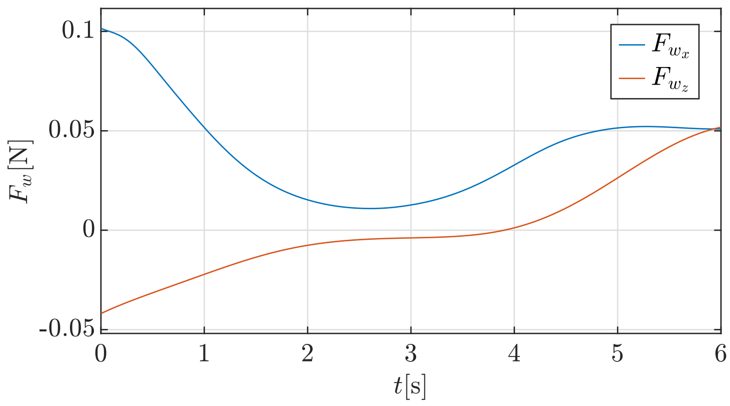


Figure 7.20: Profiles of the forces acting on the UAM in the experiments for the comparison between RAC and RMRC

Figure 7.21 presents the comparison of the accelerations of the manipulator motors. As can be seen, the trends are more or less similar with a difference at the moment of impulse. Shortly after that instant it can be seen that the accelerations required of the motors in the case where the UAM is controlled by RAC algorithm reach higher peaks.

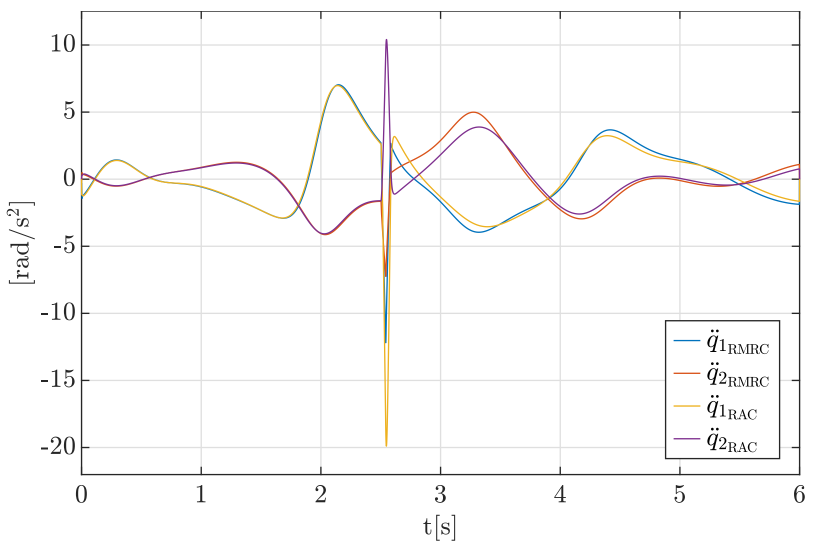


Figure 7.21: Comparison of manipulator motors acceleration profiles. UAM under the effect of the aerodynamic force plotted in 7.20 and impulsive force at $t = T_{\text{path}}/2$

Figures 7.22 e 7.23 show the comparison of linear and angular velocities in the two cases. Here again, the greatest differences occur at the time of the impulse.

Finally, Figure 7.24 shows the comparison of the EE position error norms. It would appear that the RAC reacts worse to impulsive events. However subsequently it is able to maintain the error slightly lower than the RMRC.

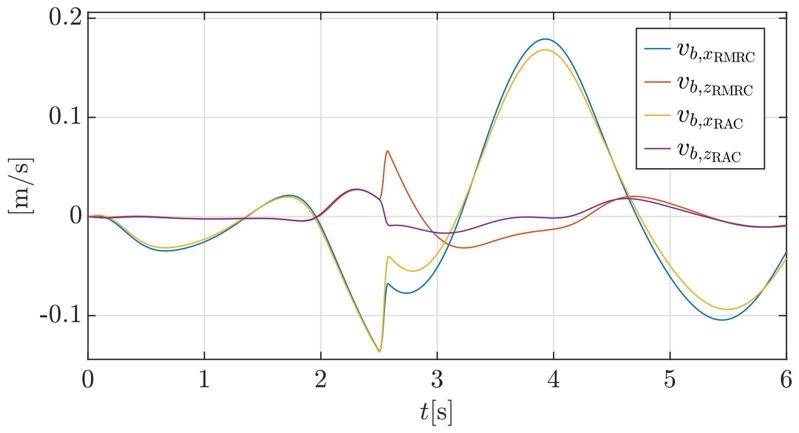


Figure 7.22: Comparison of base velocity components. UAM under the effect of the aerodynamic force plotted in 7.20 and impulsive force at $t = T_{\mathrm{path}}/2$

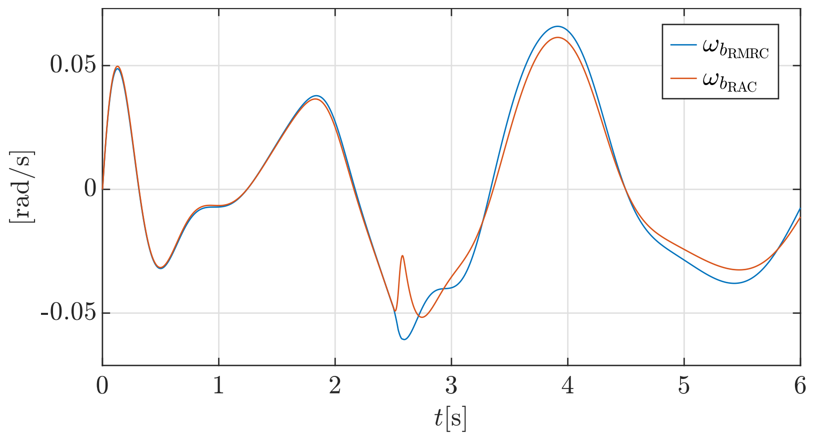


Figure 7.23: Comparison of base angular velocity. UAM under the effect of the aerodynamic force plotted in 7.20 and impulsive force at $t = T_{\mathrm{path}}/2$

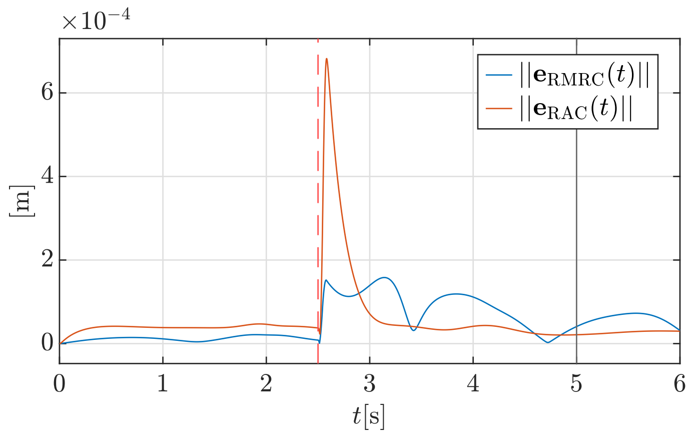


Figure 7.24: Comparison of $||\boldsymbol{e}(t)||$ in the two virtual experiments

Conclusions

In this thesis, new methods for trajectory tracking control by aerial manipulators (UAMs) were presented and validated. Initially, the mathematical model of the reference aerial platforms for the UAM was derived, namely the DJI S1000 commercial octocopter. This is an underactuated system that is supposed to be controlled through a PID system acting on altitude, roll, pitch, and yaw. Modeling and control of traditional fixed-base manipulators was then discussed, with special reference to CLIK algorithms.

Next, the equations underlying a new simulator implemented in MATLAB code for dynamic simulation of simplified planar models of UAMs were presented. The accuracy of the simulator was demonstrated by comparing the results of a benchmark virtual experiment with those coming out of the dynamic simulation software ADAMS.

To test the UAM under conditions closer to real-world ones, several models of some of the most common sources of disturbance in UAV control were implemented in this environment: forces of aerodynamic nature, noise in signals coming from sensors, and uncertainty about robot parameters.

Two methods were then presented for resolving inverse kinematics and controlling UAMs. These allow determining the references to be given to the manipulator joints for tracking a given trajectory in the operational space. The first, called Resolved Motion Rate Control (RMRC), provides velocity references while the second, called Resolved Acceleration Control (RAC), provides acceleration profiles. Both are made closed-loop (CLIK algorithms) by adding a feedback term based on the operational space error.

The final section was devoted to showing the results of several tests on the

RAC algorithm, new to UAMs, which demonstrated good rejection capability for all modelled disturbances. Also shown are the results of a preliminary sensitivity analysis conducted to decree the best combination of gains to be assigned to the system.

Possible future developments of the work done in this thesis could be the followings:

- The algorithms were tested for the control of a 2-DoFs manipulator with the objective of tracking a trajectory in the plane. In the future, these are also to be tested on redundant manipulators.

- Joint limits (both position and velocity) were not taken into account in planning and control. These affect performances in important ways and cannot be ignored in the future.

- The dynamic model of the UAM could be made more accurate if the dynamic effects related to the individual joint were also considered. For example, the mass and inertia of each servo motor and friction in the gearboxes. It was also assumed that the motors are able to perfectly follow the output references from the inverse kinematics. In reality there is a low-level feedback control at the level of the individual robot joint that has its own dynamics that may not be negligible.

- One could enrich the simulator by modeling the aerodynamic effects that could affect UAM in indoor scenarios due to proximity to walls and ground (known in the literature as wall and ground effects).

- In this thesis, the drone was considered to be controlled only in altitude and pitch by a PID system. Today there are many other methods and strategies for controlling UAVs even in horizontal motion. RMRC and RAC algorithms will have to be tested also in combination with these new methods.

Bibliography

[1] H. Bonyan Khamseh, F. Janabi-Sharifi, and A. Abdessameud, “Aerial manipulation—a literature survey,” *Robotics and Autonomous Systems*, vol. 107, pp. 221–235, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889017305535

[2] F. Ruggiero, V. Lippiello, and A. Ollero, “Aerial manipulation: A literature review,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1957–1964, 2018.

[3] A. Ollero, M. Tognon, A. Suarez, D. Lee, and A. Franchi, “Past, present, and future of aerial robotic manipulators,” *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 626–645, 2022.

[4] X. Ding, P. Guo, K. Xu, and Y. Yu, “A review of aerial manipulation of small-scale rotorcraft unmanned robotic systems,” *Chinese Journal of Aeronautics*, vol. 32, no. 1, pp. 200–214, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1000936118301894

[5] A. Pasetto, “Minimizzazione dei disturbi dinamici trasmessi ad un uav durante operazioni di manipolazione aerea,” Tesi di Laurea magistrale in Ingengeria Meccanica, Università degli Studi di Padova, 2021-2022.

[6] O. k. B. Siciliano, *Handbook of Robotics.* Springer Berlin, 2016.

[7] E. Rossetto, “Interazione dinamica tra moto del manipolatore e dell’uav nella manipolazione aerea,” Tesi di Laurea magistrale in Ingengeria Meccanica, Università degli Studi di Padova, 2019-2020.

[8] DJI, *Spreading Wings S1000+ User Manual v1.4*, January 2016.

[9] A. Sidea, R. Brogaard, N. Andersen, and O. Ravn, "General model and control of an n rotor helicopter," *Journal of Physics: Conference Series*, vol. 570, p. 052004, 12 2014.

[10] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[11] M. Orsag, C. Korpela, P. Oh, and S. Bogdan, "Aerial manipulation," 01 2018.

[12] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control.* Springer Publishing Company, Incorporated, 2010.

[13] S. Cocuzza, I. Pretto, and S. Debei, "Least-squares-based reaction control of space manipulators," *Journal of Guidance Control and Dynamics*, vol. 35, pp. 976–986, 2012.

[14] M. Wilde, S. Kwok Choon, A. Grompone, and M. Romano, "Equations of motion of free-floating spacecraft-manipulator systems: An engineer's tutorial," *Frontiers in robotics and AI*, vol. 5, p. 41, 2018. [Online]. Available: https://europepmc.org/articles/PMC7806027

[15] V. Viktor, I. Valery, A. Yuri, I. Shapovalov, and D. Beloglazov, "Simulation of wind effect on a quadrotor flight," vol. 10, pp. 1535–1538, 01 2015.

[16] S. L. Waslander and C. Wang, "Wind disturbance estimation and rejection for quadrotor position control," 2009.

[17] [Online]. Available: https://it.mathworks.com/help/aeroblks/drydenwindturbulencemodelcontinuous.html

[18] C. Menon, A. Aboudan, S. Cocuzza, A. Bulgarelli, and F. Angrilli, "Free-flying robot tested on parabolic flights: Kinematic control," *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM*, vol. 28, pp. 623–630, 07 2005.

[19] Y. Umetani and K. Yoshida, "Resolved motion rate control of space manipulators with generalized jacobian matrix," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 303–314, 1989.

[20] A. Pasetto, Y. Vyas, and S. Cocuzza, "Zero reaction torque trajectory tracking of an aerial manipulator through extended generalized jacobian," *Applied Sciences*, vol. 12, no. 23, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/23/12254

[21] R. Mukherjee and Y. Nakamura, "Formulation and efficient computation of inverse dynamics of space robots," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 400–406, 1992.