SAPIENZA
UNIVERSITÀ DI ROMA

TITOLO TESI

Facoltà di Medicina e Psicologia
Laurea Magistrale in Medicina e Chirurgia

Nome Cognome
ID number MATRICOLA

Relatrice
Dott.ssa Nome Cognome

Academic Year 2022/2023

Thesis defended on 24 Gennaio 2024
in front of a Board of Examiners composed by:

Prof. Tizio (chairman)

Prof. Caio

Prof. Sempronio

Prof. ...

Prof. ...

TITOLO TESI
Tesi sperimentale retrospettiva monocentrica. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: email autore

Abstract

Contesto

Obbiettivi

Metodi

Risultati

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Previous Works

In the context of S&R[1] operations, the support of robots has become more and more extensive in recent years [36],[14] and in particular the necessity and advantages of using *decentralized multi-agent* systems [22]. A particular field of S&R missions focuses on high mountain scenarios, in which UAVs [2] are tasked with localizing avalanche victims. These missions involve the smart collaboration between UAVs and humans, developed also in the context of European founded projects such as the SHERPA one [20].

Different types of sensors can be used to achieve the localization goal, but the fastest and most accurate ones are electromagnetic sensors, which can operate also in noisy environments [29]. Among these, the ARTVAs[3] are the most commonly used in both human-conducted and robot-conducted operations.

Many studies have analyzed and formulated strategies and algorithms in the particular context of S&R operations using UAVs for avalanche victims [44], [40], [43]. Furthermore, in order to address the localization problem, the authors of [10] and [9] propose a robocentric SLAM approach in the broader robotics context. Note that in [10] they also take into consideration the *multiple victims* case.

Avalanche victims rescue missions are particularly difficult with respect to other S&R operations because of some challenging aspects. In particular, the time constraint is quite demanding since the survival chances of avalanche victims diminish rapidly with time. Victims buried under avalanches have a 93% survival rate within the first 15 minutes, which drops to 25% after 45 minutes due to hypothermia [2] and plateaus at 21% from 60 min to 180 minutes [41].

The common human-conducted avalanche victims S&R technique using ARTVA technology relies on three phases. In the first phase, rescuers search for the first valid electromagnetic signal, which can be detected at distances ranging from 20 meters for single-antenna receiver to 80 meters for triple-antenna ones. In the second phase, rescuers are trained to interpret the magnetic field data and follow standard procedures in order to follow magnetic field direction and find the victim's position. In the final phase, rescuers dig to save the buried individual.

[1] S&R stands for Search and Rescue.

[2] UAV stands for Unmanned Aerial Vehicles.

[3] ARTVA stands for the Italian "Apparecchio di Ricerca dei Travolti in VAlanga".

Despite the effectiveness of this technique, it requires a significant amount of time due to the challenges of traversing avalanche terrain. Additionally, rescuers walking on unstable snow face the tangible risk of triggering a secondary avalanche [9]. For these reasons, and given the additional advantage of typically not encountering obstacles in high mountain scenarios, the use of intelligent *autonomous* drones results in a faster and safer search when compared to human rescuers, as shown in [27], [33], [32].

Therefore, the aim of this work is to build upon previous efforts to *automate* and improve the efficiency of the second phase by developing a mathematical/algorithmic framework for solving the localization problem of not just one, but *multiple avalanche victims*, using *multiple decentralized* agents (UAVs). This approach aims to reduce computational complexity without sacrificing accuracy and convergence time.

1.1.1 Magnetic Dipole Localization

In the context of a single magnetic source, numerous studies have focused on the localization task; in [39] a straightforward closed-form solution is found by leveraging the analytical expressions of the magnetic field vector and the magnetic gradient tensor, regardless of the singularity of the magnetic gradient tensor matrix.

In the field of TMI[4] gradiend surveys, the acquisition of magnetic gradinet tensor data has seen significant advancements. In [16], new methods have been developed for inverting gradient tensor surveys, for a number of elementary, but useful, models. These include point pole, line of poles, point dipole (sphere), line of dipoles (horizontal cylinder), thin and thick dipping sheets, sloping step, and contact models. A key insight is the use of eigenvalues and associated eigenvectors of the magnetic tensor to obtain the NSS [5], which is a particularly useful rotational invariant that peaks directly over 2D sources, and it is independent of magnetization direction.

The NSS has been employed not only in geological applications, but also in the the detection, location, and classification of magnetic objects, such as naval mines, UXO, shipwrecks, archaeological artifacts, and buried drums [16].

In [47], the authors use the spatial relation between the magnetic target and observation points derived from the NSS for accurately locating a magnetic target, achieving nearly error-free results in the absence of noise.

In addition, methods that use the NSS for magnetic sources localization, for different models and dimensions is done in [17].

In [52], a key aspect of the NSS is identified by the fact that it is only dependent on the distance between the source and target position; since an analytical expression of the magnetic gradient tensor and a closed-form localization formula are derived.

In [15] as well, it is stressed that the NSS is proportional to a constant normalized by the distance between observation and source. It is independent of magnetization direction and satisfies Euler's homogeneity equation, this allows the authors to use Euler deconvolution of the NSS to estimate source location.

Instead, the more specific multi-magnetic source localization task has been a difficult problem and different strategies have been conducted.

[4] TMI stands for Total Magnetic Intensity

[5] NSS stands for Normalized Source Strength.

In recent years, many studies have performed an in-depth analysis on rotational invariants of the magnetic gradient tensor, such as [34] and [53]. In the latter, a new method, named NED[6] has been proposed, which is free from geomagnetic interference and provides abundant information, and it is compared with other methods such as the tilt angle (ratio of vertical to horizontal magnetic field components), theta map, etc.

In this context of detecting and localizing multiple dipole-like magnetic sources, the authors of [45] use the magnetic gradient tensor data: the tilt angle is used to determine the number of sources, while the rotational-invariant NSS is used to estimate the horizontal coordinates. Then, they employ the DE algorithm estimates the locations and moments of the sources.

Lastly, the authors of [37] use the principal invariants of the magnetic field gradient tensor to determine the number and horizontal location of the magnetic sources, while Euler equations are used to compute the sources's depth.

In our work, between all the different indicators in literature, we chose to employ the NSS in order to estimate the horizontal coordinates of the avalanche victims, since it is rotationally invariant and completely isotropic around the magnetic dipole. We are not interested in estimating the burial depth of the victims, even if [8] use a closed loop formula to find z, it cannot be applied to the multiple sources case.

1.1.2 Particle Swarm Optimization

In this context of simultaneous localization of multiple magnetic dipole sources, like in this real world application [5]; different algorithms have been employed. For instance, in [38], the authors use a non-linear optimization method based on the Levenberg–Marquardt algorithm to solve the problem, without the prior knowledge of the number of dipoles in the 3-D detection region.

In this work, we propose the use of the PSO[7] algorithm as a mean to solve the multiple-sources localization task, which can be considered a continous multi-modal optimization problem.

The PSO[8] algorithm was first introduced in 1995 [3], and it has since proven to be a simple but powerful tool for solving various optimization problems[28], among a lot of EAs [9]. It has been used to solve various practical applications like simultaneous localization of multiple odor souces [26], as well as detection of acustic [12], thermal, or chemical/biological signals [31]. In the robotics field, it has not only been employed in source-seeking applications but aslo path-planning scenarios [48] and deployment of communication networks [46].

In the standard PSO algorithm [3], a population of particles is randomly initialized to represent potential candidate solutions for the optimization problem. Each particle relies on two important pieces of information: its individual best solution, referred to as *pbest*, and the global best across the entire population, referred to as *gbest*. These two values guide the search direction of all particles over the search space. The evaluation of the *pbest* for each particle and the *gbest* for the entire population is determined by the function that needs to be optimized.

[6] NED stands for NEw Edge Detection
[7] PSO stands for Particle Swarm Optimization.
[8] PSO stands for Particle Swarm Optimization.
[9] EA stands for Evolutionary Algorithm.

However, when addressing continuous optimization problems with numerous local optima, classical optimization algorithms often require strict conditions and extensive computation times [50]. For this reason, the authors of [28] introduced a modified PSO algorithm where the original population is divided into multiple subpopulations, according to the order of particles. The best particle within each subpopulation is recorded and then used into the velocity updating formula to replace the original global best particle of thge entire population. This strategy, based on the idea of multiple subpopulations, enables the algorithm to find several optima, including both the global and local solutions, thanks to these distinct best particles in each group.

Both in the standard PSO [3] and the multi-swarm modification [28] common limitations are the premature convergence of the algorithm and its tendency to become trapped in some local optima. Instead in [50], the authors overcome these issue, including the one of poor population diversity. They introduce a modification to [28], it is called ADPSO[10], and includes features for stagnation detection and spatial exclusion to mitigate these challenges. The dynamic division of the population into sub-swarms, coupled with a regrouping mechanism, avoids stagnation during the optimization process. Stagnant sub-swarms are rejuvenated with a new generate particle, called the vitality particle, constructed from the historical information of the entire population. This approach maintains diversity and prevents sub-swarms from converging prematurely.

The authors of [30] focus their attention to the exploration and exploitaion aspects of the PSO algorithm, instead. They enhance exploration and exploitation in a modified system, named HCLPSO [11], which divides the population into only two sub-swarms, one dedicated to exploitation and one dedicated to exploration. In the exploration-subpopulation, the solutions are generated by using the personal best experiences of the particles in the exploration-subpopulation itself. In the exploitation-subpopulation, the personal best experiences of the entire swarm population are used to generate the solutions. In this way, since the exploration-subpopulation does not learn from any particles in the other subpopulation, diversity is mantained in the exploration-subpopulation even when the exploitation-subpopulation converges prematurely.

1.2 The ARTVA

The ARTVA technology is composed of two different and easy switchable modalities: in *receiver* mode, the instrument senses and processes the electromagnetic field emitted by the ARTVA *transmitter* (carried by the avalanche victim).

The magnetic field generated by the solenoid antenna of the instrument oscillates with a frequency of 457 kHz and its characteristics are defined in the standard ETS 300 718-1 [42], to ensure compatibility between different brands and models. To save batteries and facilitate detection, the magnetic field is transmitted in pulses of a tenth of a second every second [9].

As will be discussed more in-depth in the following chapter, the magnetic field can be modeled as a three-dimensional vector field, which means that it assigns a certain intensity and direction to each point in space. Therefore, the main difference between different kinds of ARTVAs lies in *"how much"* of this field they can measure. According to this criterion, the instruments can be divided into three different types [9]:

[10] ADPSO stands for Adaptive Dynamic Multi-Swarm Particle Swarm Optimization.

[11] HCLPSO stands for Heterogeneous Comprehensive Learning Particle Swarm Optimization.

- ARTVAs *with one reception antenna*: the oldest models, usually analog. The same antenna is used in both *transmitter* and *receiver* mode. Therefore, only the projection of the magnetic field on the antenna can be measured. This type of ARTVA is the most difficult to use and the most time-consuming one.
- ARTVAs *with two perpendicular reception antennas*: are based on digital technology such as microprocessors. This type can measure only the intensity and direction of the horizontal component of the field, only when held in horizontal position.
- ARTVAs *with three mutually perpendicular reception antennas*: also based on digital technology. Since these ARTVAs possess three perpendicular antennas, they can measure the complete vector field. For this reason, the instrument can be oriented w.r.t the magnetic field in any way.

In this work, we will consider only new ARTVA transceivers (three antennas), which can achieve a search strip width in digital mode of 80 m and a maximum range in analog mode of 90 m [51]. Furthermore, it is important to point the transceiver in the direction of the avalanche, parallel to the slope. For this reason, in this work, two different types of trajectories have been considered.

1.3 Gradient, Divergence, and Curl

We briefly define the following operators which will be used throughout this work.

1.3.1 Gradient of a Scalar Field

Definition

Given a scalar function $f(x_1, x_2, \ldots, x_n) : \mathbb{R}^n \to \mathbb{R}$ or scalar field, the gradient of the function is a vector field of partial derivatives and denoted by ∇f, is defined as:

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3}, \ldots, \frac{\partial f}{\partial x_n} \right) \tag{1.1}$$

where ∇ is the vector differential operator [21].

Properties

The gradient is a vector which points in the direction of the greatest ascent of the function and its magnitude is the increase.

1.3.2 Definition of a Vector Field

A **vector field** on a subset $S \subseteq \mathbb{R}^n$ is a vector-valued function $\mathbf{V} : S \to \mathbb{R}^n$ that assigns to each point $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in S$ a vector $\mathbf{V}(\mathbf{x})$ [21].
Note that from now on, we usually omit the spatial dipendence of vector fields from points $\mathbf{x}$, since it's implied.

1.3.3 Divergence of a Vector Field

For simplicity and since we are working with $\mathbb{R}^3$ Euclidean space, we limit our discussion from now on to Euclidean coordinates.

Definition

Given a vector field $\mathbf{V}$, the divergence of $\mathbf{V}$ at a point $\mathbf{p} \in \mathbb{R}^3$ is defined as the net outward flux of $\mathbf{V}$ per unit volume Δv as the volume about the point tends to zero:

$$\nabla \cdot \mathbf{V} = \lim_{\Delta v \to 0} \frac{1}{\Delta v} \iint_S \mathbf{V} \cdot d\mathbf{s} \tag{1.2}$$

where $\mathbf{V} \cdot d\mathbf{s}$ is the flux of $\mathbf{V}$ through the surface S [23].

Since $\mathbf{V} = V_x \mathbf{e_x} + V_y \mathbf{e_y} + V_z \mathbf{e_z}$, the divergence of $\mathbf{V}$, can be computed as:

$$\nabla \cdot \mathbf{V} = \frac{\partial V_x}{\partial x} + \frac{\partial V_y}{\partial y} + \frac{\partial V_z}{\partial z} \tag{1.3}$$

where V_x, V_y, and V_z are the components of the vector field $\mathbf{V}$ in the x, y, and z directions, respectively [21].

Properties

When the vector field is represented using flux lines (indicating the direction and intensity), the divergence is the amount of flux lines diverging/converging through a given point.

A net outward flux of a vector field through a surface bounding a volume indicates the presence of a source, the divergence measures the strength of the source. The divergence of a vector field is a scalar field.

1.3.4 Laplacian of a Vector Field

The Laplacian of a vector field $\mathbf{V}$, denoted by ∇^2, is similar to the scalar Laplacian and is defined as:

$$\nabla^2 V = \frac{\partial^2 V_x}{\partial x^2} + \frac{\partial^2 V_y}{\partial y^2} + \frac{\partial^2 V_z}{\partial z^2} \tag{1.4}$$

Alternatively, by taking the curl of the curl of a vector field, the Laplacian can be expressed as:

$$\nabla^2 \mathbf{V} = \nabla(\nabla \cdot \mathbf{V}) - \nabla \times (\nabla \times \mathbf{V}) \tag{1.5}$$

1.3.5 Curl of a Vector Field

Definition

The curl of $\mathbf{V}$, denoted by $\nabla \times \mathbf{V}$, at a point in space $\mathbf{x} \in \mathbb{R}^3$ is a vector field whose magnitude is the maximum net circulation of $\mathbf{V}$ per unit area as the area tends to zero and whose direction is the normal direction of the area when the area is oriented to make the net circulation maximum:

$$\nabla \times \mathbf{V} = \lim_{\Delta s \to 0} \frac{1}{\Delta s} \mathbf{n} \oint_C \mathbf{V} \cdot d\mathbf{l} \tag{1.6}$$

where $\mathbf{n}$ is the unit normal vector to the surface S, $d\mathbf{l}$ is the differential line element along the boundary, and the integral represents the circulation of $\mathbf{V}$ around the boundary of the surface [23].

The curl of a vector field $\mathbf{V}$ can be then computed in terms of its components [21]:

$$\nabla \times \mathbf{V} = \left(\frac{\partial V_z}{\partial y} - \frac{\partial V_y}{\partial z}, \frac{\partial V_x}{\partial z} - \frac{\partial V_z}{\partial x}, \frac{\partial V_y}{\partial x} - \frac{\partial V_x}{\partial y} \right) \tag{1.7}$$

Properties

Since the normal to an area can point in two opposite directions, the direction of the curl is given by the right-hand rule. A vortex source causes a circulation of the vector field around it. The circulation of a vector field around a closed path is defined as the scalar line integral of the vector over the path. Note that a circulation of $\mathbf{V}$ can exist even when the divergence of $\mathbf{V}$ is zero, meaning there is no net source or sink.

1.4 Stokes' Theorem

The surface integral of the curl of a vector field $\mathbf{V}$ over a surface s is equal to the line integral of the vector field over the boundary contour c of the surface:

$$\oint_C \mathbf{V} \cdot d\mathbf{l} = \iint_S (\nabla \times \mathbf{V}) \cdot d\mathbf{s} \tag{1.8}$$

The proof of the theorem comes directly from the definition of the curl 1.6 and by dividing the surface S into smaller areas. The idea comes from the fact that computing the line integral around the boundary of a surface is equal to compute the integral for all the smaller areas, since the $d\mathbf{l}$ components of the neighbouring regions are in opposite directions.

1.5 Divergence Theorem

The theorem states that the surface integral of a vector field $\mathbf{V}$ over a closed surface S is equal to the volume integral of the divergence of $\mathbf{V}$ over the volume enclosed by S:

$$\iint_S \mathbf{V} \cdot d\mathbf{s} = \iiint_V (\nabla \cdot \mathbf{V})\, dv \tag{1.9}$$

The idea of the proof is similar to 1.4, starting from the definition of the divergence 1.2. Considering a volume divided into smaller volumes, the contributions from the internal surfaces cancel each other, leaving only the contribution from the outer surface.

1.6 Null Identity Theorem

The divergence of the curl of any vector field is always zero [23]:

$$\nabla \cdot (\nabla \times \mathbf{V}) = 0 \tag{1.10}$$

The proof leverages the divergence theorem 1.9 applied to the vector field $\nabla \cdot (\nabla \times \mathbf{V})$. Considering that any volume can be divided in half, then the surface bounding the volume

would be the sum of 2 surfaces, connected by a common boundary that has been drawn twice. One can then compute the two surface integrals using 1.8, and since the two normals $\mathbf{n}$ have equal intensity and opposite direction, their sum is zero and the integrand as well.

A converse statement of the theorem is as follows: If a vector field $\mathbf{B}$ is divergence-less, then it can be expressed as the curl of another vector field $\mathbf{V}$:

$$\nabla \cdot \mathbf{B} = 0 \implies \mathbf{B} = \nabla \times \mathbf{V}$$

Since the identity $\nabla \cdot (\nabla \times \mathbf{V}) = 0$ always holds, it means that for any magnetic field $\mathbf{B}$ with zero divergence, we can find a vector potential $\mathbf{V}$ such that $\mathbf{B}$ is the curl of $\mathbf{V}$.

1.7 Phasor Notation

For dealing with time-varying vector fields, we use phasor notation to represent sinusoidal varying field vectors (which is usually the case in real-world applications, such as the magnetic dipole) [23]. Phasor notation simplifies the analysis of such fields by converting differential equations into algebraic equations.

Then, an harmonic vector field $\mathbf{V}(x, y, z, t) = \mathbf{V}(x, y, z) \cos \omega t$ can be represented by a vector phasor that depends on space coordinates but not on time:

$$\mathbf{V}(x, y, z, t) = \mathbf{V}(x, y, z)e^{j\omega t}$$

where ω is the angular frequency and $\mathbf{V}(x, y, z)$ is a vector phasor that contains information on direction, magnitude, and phase.

The time-domain function can be recovered from the phasor by taking the real part:

$$\mathbf{V}(x, y, z, t) = \mathrm{Re}\{\mathbf{V}(x, y, z)e^{j\omega t}\}$$

1.8 Spherical Coordinates

We define two classical coordinate systems: spherical coordinates (r, θ, ϕ) and Cartesian coordinates (x, y, z), with their respective vector basis $\{\mathbf{e}_r, \mathbf{e}_\theta, \mathbf{e}_\phi\}$ and $\{\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z\}$, represented in Figure 1.1, from which the following relationships are evident. Note that we use the convention in [44] for the angles and the axes.

1.8.1 Conversion from Cartesian to Spherical Coordinates

$$r = \sqrt{x^2 + y^2 + z^2}, \tag{1.11}$$

$$\theta = \cos^{-1}\left(\frac{\sqrt{z^2 + y^2}}{x}\right), \tag{1.12}$$

$$\phi = \tan^{-1}\left(\frac{y}{z}\right). \tag{1.13}$$

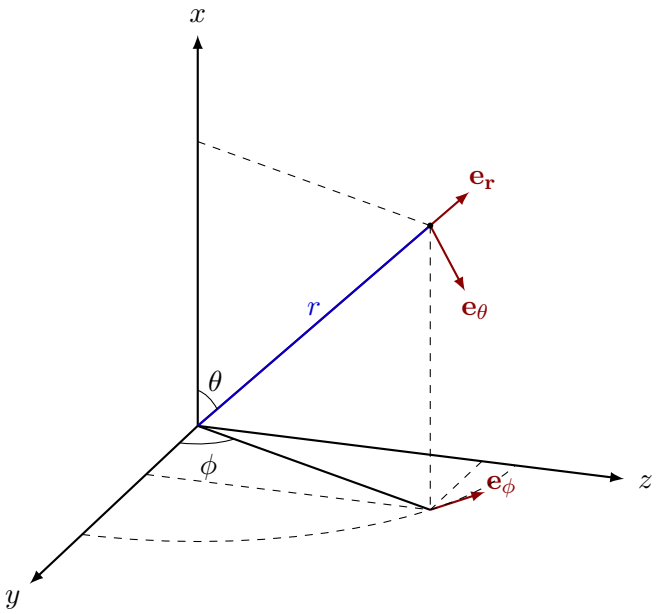

Figure 1.1 Spherical and Cartesian coordinates with their respective unit vectors.

1.8.2 Conversion from Cartesian to Spherical Coordinates

$$x = r\cos\theta, \tag{1.14}$$
$$y = r\sin\theta\cos\phi, \tag{1.15}$$
$$z = r\sin\theta\sin\phi. \tag{1.16}$$

1.8.3 Expressing Spherical Unit Vectors using Cartesian Unit Vectors

A point $\mathbf{r}$ in Cartesian coordinates is given by:

$$\mathbf{r} = x\,\mathbf{e}_x + y\,\mathbf{e}_y + z\,\mathbf{e}_z.$$

The radial unit vector $\mathbf{e}_r$ is defined as the normalized position vector, obtained by substituting the coordinates in 1.8.1:

$$\mathbf{e}_r = \frac{\mathbf{r}}{|\mathbf{r}|} = \cos\theta\,\mathbf{e}_x + \sin\theta\cos\phi\,\mathbf{e}_y + \sin\theta\sin\phi\,\mathbf{e}_z. \tag{1.17}$$

The unit vector $\mathbf{e}_\theta$, which is perpendicular to $\mathbf{e}_r$ and lies in the plane formed by the origin and the z-axis, points in the direction of increasing θ. It can be found by taking the partial derivative of the position vector with respect to θ and normalize it:

$$\mathbf{e}_\theta = -\sin\theta\,\mathbf{e}_x + \cos\theta\cos\phi\,\mathbf{e}_y + \cos\theta\sin\phi\,\mathbf{e}_z. \tag{1.18}$$

Similarly, the unit vector $\mathbf{e}_\phi$, which is perpendicular to both $\mathbf{e}_r$ and $\mathbf{e}_\theta$, points in the direction of increasing ϕ. It can be derived by taking the partial derivative of the position vector with respect to ϕ and normalize it:

$$\mathbf{e}_\phi = \frac{\frac{\partial \mathbf{r}}{\partial \phi}}{|\frac{\partial \mathbf{r}}{\partial \phi}|} = -\sin\phi\,\mathbf{e}_y + \cos\phi\,\mathbf{e}_z.$$

1.8.4 The Gradient, Divergence, and Curl in Spherical Coordinates

We omit the demonstration of how these formulas are found starting from the definitions given in Cartesian coordinates 1.3, which involve large amount of computations [23].

For a vector field expressed in spherical coordinates:

$$\mathbf{V} = V_r \mathbf{e}_R + V_\theta \mathbf{e}_\theta + V_\phi \mathbf{e}_\phi$$

the divergence is:

$$\nabla \cdot \mathbf{V} = \frac{1}{r^2}\frac{\partial}{\partial r}(r^2 V_r) + \frac{1}{r\sin\theta}\frac{\partial}{\partial \theta}(\sin\theta V_\theta) + \frac{1}{r\sin\theta}\frac{\partial V_\phi}{\partial \phi}. \tag{1.19}$$

The curl in spherical coordinates is:

$$\nabla \times \mathbf{V} = \frac{1}{r\sin\theta}\begin{vmatrix} \mathbf{e}_r & r\mathbf{e}_\theta & r\sin\theta\mathbf{e}_\phi \\ \frac{\partial}{\partial r} & \frac{\partial}{\partial \theta} & \frac{\partial}{\partial \phi} \\ V_r & rV_\theta & r\sin\theta V_\phi \end{vmatrix}. \tag{1.20}$$

carrying out the determinant:

$$\begin{aligned} \nabla \times \mathbf{V} = \frac{1}{r\sin\theta}\Bigg[&\left(\frac{\partial}{\partial\theta}(V_\phi \sin\theta) - \frac{\partial V_\theta}{\partial \phi}\right)\mathbf{e}_r \\ &+ \left(\frac{1}{\sin\theta}\frac{\partial V_r}{\partial \phi} - \frac{\partial}{\partial r}(rV_\phi)\right)\mathbf{e}_\theta \\ &+ \left(\frac{\partial}{\partial r}(rV_\theta) - \frac{\partial V_r}{\partial \theta}\right)\mathbf{e}_\phi\Bigg]. \end{aligned} \tag{1.21}$$

1.9 3D Dirac Delta

Definition

Given a point $\mathbf{r}$ in Cartesian coordinates, the Dirac delta function $\delta(\mathbf{r})$ is defined as [24]:

- $\delta(\mathbf{r}) = 0$ at all points except at $\mathbf{r} = (0, 0, 0)$.
- The integral across the entire space satisfies:

$$\int_V \delta(\mathbf{r})\, dv = 1 \tag{1.22}$$

The result of the integral could be the value of any function in zero, $f(0)$. For example, the Dirac delta could be the charge density ρ 2.1 of a point particle located at the origin whose charge is q.

1.10 Green's Function for Poisson Equation

Poisson's equation using Green's function is written as:

$$\nabla^2 G(\mathbf{r}) = \delta(\mathbf{r}) \tag{1.23}$$

The solution to the above equation is given by:

$$G(r) = -\frac{1}{4\pi r}$$

Proof

We assume $G(r)$ to be axis-symmetric, implying that it only depends on the magnitude r, not the vector position $\mathbf{r}$. Considering the point (x, y, z) and the distance r from the origin. To find Green function we look for the simplest function that satisfies the equation, which has the form:

$$G(r) = A\frac{1}{r} + B$$

where A and B are constants.

Assuming $B = 0$ for simplicity, we find A by integrating over a sphere of volume ϵ. Substituting Poisson's equation 1.23 in the the definition of the Dirac delta 1.22:

$$\int_V \nabla^2 G(r)\, dv = 1$$

Using the divergence theorem 1.9:

$$\int_V \nabla^2 G(r)\, dv = \int_S \nabla G(r) \cdot d\mathbf{s}$$

$$1 = \int_S \nabla G(r) \cdot d\mathbf{s}$$

The right-hand side represents the flux through the surface S. We take the integral over the surface of a sphere of radius r, knowing that the surface area is $4\pi r^2$ and computing the divergence (which is just the derivative thanks to the assumption) of $A\,\frac{1}{r}$:

$$1 = \int_S -\frac{A}{r^2} d\mathbf{s} = -\frac{4\pi r^2 A}{r^2}$$

From which we conclude:

$$A = -\frac{1}{4\pi}$$

Chapter 2

Electromagnetism

The ARTVA instrument in transmitter and receiver mode is a magnetic dipole. In order to formulate a coherent mathematical model, it is necessary to report some results of electromagnetic theory. Firstly, we identify and name the fundamental electromagnetic physical quantities:

Table 2.1 Electromagnetic Physical Quantities

Symbol	Description	Unit
$\mathbf{E}$	Electric field intensity	V/m
$\mathbf{D}$	Electric displacement field	C/m^2
$\mathbf{H}$	Magnetic field intensity	A/m
$\mathbf{B}$	Magnetic flux density	T
$\mathbf{J}$	Current density	A/m^2
$\mathbf{A}$	Magnetic vector potential	V $\cdot$ s/m
$\mathbf{m}$	Magnetic vector moment	A $\cdot$ m^2
ρ	Volume charge density	C/m^3
ϵ	Permittivity of the medium	F/m
μ	Permeability of the medium	H/m
c	Speed of light in vacuum	m/s

2.1 Maxwell's Equations

We postulate Maxwell's equations in a simple (linear, isotropic, and homogeneous) medium in phasor notation 1.7, which have been discovered experimentally [23]:

$$\nabla \times \mathbf{E} = -j\omega\mathbf{B} \tag{2.1}$$
$$\nabla \times \mathbf{H} = \mathbf{J} + j\omega\epsilon\mathbf{E} \tag{2.2}$$
$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon} \tag{2.3}$$
$$\nabla \cdot \mathbf{B} = 0 \tag{2.4}$$

In these equations, the space-coordinate arguments have been omitted for simplicity. The fact that the same notations are used for the phasors as are used for their corresponding time-dependent quantities should create little confusion because we will deal exclusively with sinusoidal vector fields.

From Maxwell's equation 2.4, we know that the magnetic flux density $\mathbf{B}$ is solenoidal (zero divergence). Then, $\mathbf{B}$ can be expressed as the curl of another vector field using the Null Theorem 1.10, obtaining 2.4:

$$\mathbf{B} = \nabla \times \mathbf{A} \tag{2.5}$$

Also, $\mathbf{B}$ relates to the magnetic field intensity $\mathbf{H}$ through the permeability of the medium μ:

$$\mathbf{B} = \mu\mathbf{H} \tag{2.6}$$

Another useful form of Maxwell's first equation 2.1 can be found by substituting 2.5:

$$\nabla \times \mathbf{E} = -j\omega(\nabla \times \mathbf{A}) = -\nabla \times j\omega\mathbf{A}$$
$$\nabla \times (\mathbf{E} + j\omega\mathbf{A}) = 0$$

Since the sum of vector fields is itself a vector field, $\mathbf{E} + j\omega\mathbf{A}$ is a vector field, and we can define a **scalar** field, the electric potential V, such that:

$$\mathbf{E} + j\omega\mathbf{A} = -\nabla V \tag{2.7}$$

If the curl of a vector field is zero, a scalar field exists whose gradient gives the vector field.

2.1.1 Conservation of Charge Principle

The principle of conservation of charge states that the net charge within a closed system remains constant over time, meaning no charge can be created nor destroyed [23]:

$$\nabla \cdot \mathbf{J} = -\frac{\partial\rho}{\partial t} \tag{2.8}$$

The current density $\mathbf{J}$ is defined as:

$$\mathbf{J} = Nq\mathbf{u} \tag{2.9}$$

where N is the number of charge carriers per unit volume, q is the charge of each carrier, and $\mathbf{u}$ is the drift velocity of the charge carriers.

This means that if a current flows out of a volume, the charge density inside the volume must decrease at a rate equal to the current. The current leaving the volume is the flux of the current density through surface S:

$$I = \oint_S \mathbf{J} \cdot d\mathbf{s} \tag{2.10}$$

2.2 Wave Equation for Magnetic Vector Potential

In order to determine the intensity of the magnetic field, we first need to find an expression for the magnetic vector potential $\mathbf{A}$, called the wave equation. Starting from Maxwell's equations, we find the wave equation by substituting 2.5 and 2.6 into the second Maxwell equation 2.2:

$$\nabla \times \nabla \times \mathbf{A} = \mu\mathbf{J} + j\omega\epsilon\mu\mathbf{E}$$

Then we substitute 2.7 for $\mathbf{E}$ and use the Laplacian 1.5 on the left side:

$$\nabla(\nabla \cdot \mathbf{A}) - \nabla^2\mathbf{A} = \mu\mathbf{J} + j\omega\epsilon\mu\left(-\nabla V - j\omega\mathbf{A}\right)$$

The definition of a vector requires the specification of both its curl and its divergence. Although the curl of $\mathbf{A}$ is designated $\mathbf{B}$ in 2.5, we are still at liberty to choose its divergence to simplify the expression [23]:

$$\nabla \cdot \mathbf{A} = -j\omega\epsilon\mu V$$

Finally, rearranging the terms and substituting the square of j, we get:

$$\nabla^2\mathbf{A} + \omega^2\epsilon\mu\mathbf{A} = -\mu\mathbf{J}$$

This is the wave equation for the magnetic vector potential $\mathbf{A}$:

$$\nabla^2\mathbf{A} - k^2\mathbf{A} = -\mu\mathbf{J} \tag{2.11}$$

where $k = \omega\sqrt{\mu\epsilon}$ is the wave number, which characterizes the propagation of the electromagnetic wave in the medium.

2.2.1 Finding the Potential by Solving the Wave Equation

Since both $\mathbf{A}$ and $\mathbf{J}$ are vector fields, the wave equation 2.11 can be written for each component of $\mathbf{A}$:

$$\nabla^2 \begin{pmatrix} A_x \\ A_y \\ A_z \end{pmatrix} = k^2 \begin{pmatrix} A_x \\ A_y \\ A_z \end{pmatrix} - \mu \begin{pmatrix} J_x \\ J_y \\ J_z \end{pmatrix} \tag{2.12}$$

In order to solve 2.12, we can use the Green function for Poisson's equation 1.23 for each of the components when $k = 0$ (the case of static fields). Then, the solution is given by the formula:

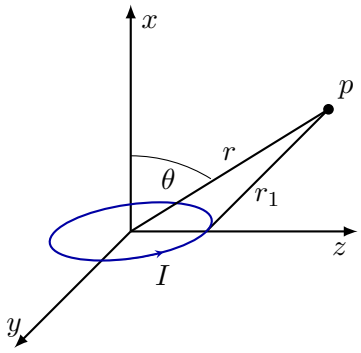

Figure 2.1 Magnetic dipole representation

$$\mathbf{A} = \frac{\mu}{4\pi} \int_V \mathbf{J} \frac{e^{-jkr}}{r} dv \tag{2.13}$$

by using the superposition principle and by finding the solution of the time-dependent differential equation.

2.3 Magnetic Dipole

We have a small filament loop of radius b, carrying an AC current $I(t) = I\cos(\omega t)$ as shown in Figure 2.1. If S is the cross-section area of the wire and dl a differential length, we have $\mathbf{J} \perp \mathbf{s}$, the normal to the area and the volume:

$$dv = S\, dl \tag{2.14}$$

In order to determine the magnetic field intensity $\mathbf{H}$ at a certain point in space p, we need to compute the magnetic vector potential $\mathbf{A}$ first [23]. Since the charges move only in the thin wire, they are located only in the wire region, then from the definition of current 2.10:

$$I = S\,\mathbf{J}$$

The volume integral in 2.13 becomes:

$$\int_V \mathbf{J}\, dv = \int_V \mathbf{J}\, S\, dl$$

Then, we substitute 2.10 and change the integral type since it is now the sum over the length, so the formula becomes (the current must flow in a closed path):

$$\mathbf{A} = \frac{\mu_0 I}{4\pi} \oint_C \frac{d\mathbf{l}}{r_1} e^{-jkr_1} \tag{2.15}$$

where r_1 is the distance between the point p and the charges (source element) and $d\mathbf{l}$ is a vector tangent to the loop of differential length dl.

Assumption

We can then simplify 2.15 by considering the radius b to be small enough, such that $r_1 - r \approx 0$. Then by adding and subtracting r from the power of the exponential:

$$e^{-jkr_1} = e^{-jk(r_1+r-r)} = e^{-jkr}\, e^{-jk(r_1-r)}$$

Then by using Taylor approximation on the second exponential ($x = r_1 - r \approx 0$) we obtain:

$$e^{-jkr_1} = e^{-jkr}\, [1 - jk(r_1 - r)]$$

Then, we substitute this result in 2.15 and simplify:

$$\begin{aligned}\mathbf{A} &= \frac{\mu_0 I}{4\pi} e^{-jkr}\, [1 - jk(r_1 - r)] \oint_C \frac{d\mathbf{l}}{r_1} \\ &= \frac{\mu_0 I}{4\pi} e^{-jkr} \left(\oint \frac{d\mathbf{l}}{r_1} - jk \oint (r_1 - r) \frac{d\mathbf{l}}{R_1} \right)\end{aligned}$$

Since the integral of $d\mathbf{l}$ over a closed loop is zero, because we have considered a small loop $b \to 0$:

$$\oint_C d\ell = 2\pi b \quad \to \quad \oint_C d\ell \to 0$$

Then we obtain:

$$\mathbf{A} = \frac{\mu_0 I}{4\pi} e^{-jkr} \left[(1 + jkr) \oint \frac{d\mathbf{l}}{r_1} \right] \tag{2.16}$$

Assumption Quasi-Static Field/Near Field Zone

If we consider a region near the magnetic dipole, we obtain quasi-static fields. We defined the wave number k as:

$$k = \omega\sqrt{\mu\epsilon} \tag{2.17}$$

Electromagnetic waves propagate with velocity u (speed of light in vacuum) [23]:

$$u = \frac{1}{\sqrt{\mu\epsilon}} \tag{2.18}$$

Then, by inverting 2.18 and substituting in 2.17, we can write k as:

$$k = \frac{\omega}{u} \tag{2.19}$$

From wave theory $f = \frac{\omega}{2\pi}$ and $\lambda = \frac{u}{f}$, we obtain another expression for u:

$$u = \frac{\lambda\, \omega}{2\, \pi} \tag{2.20}$$

Therefore, we can substitute 2.3 in 2.19:

$$k = \frac{2\pi}{\lambda} \tag{2.21}$$

To simplify the expression for $\mathbf{A}$ 2.15, we make the assumption that $k\,r \ll 1$, and if we substitute the found expression of k 2.21:

$$k\,r \ll 1 \implies \frac{2\pi r}{\lambda} \ll 1 \implies r \ll \frac{\lambda}{2\pi}$$

This means that r needs to be small in comparison to λ. If this is the case:

$$e^{-jkr} \approx e^{0} = 1$$

We eliminate completely the time dependence and obtain the expression for $\mathbf{A}$:

$$\mathbf{A} = \frac{\mu_0 I}{4\pi} \oint_C \frac{d\mathbf{l}}{r_1} \tag{2.22}$$

In the ARTVA case, the standard operating frequency is $f = 475\,\text{kHz}$ and the optimal range of the instrument is $< 80\,\text{m}$. Then in the worst case, when $r = 80\,\text{m}$, we obtain the approximation $k\,r = 0.79$.

Symmetry

In the particular case of a magnetic dipole, the magnetic vector potential $\mathbf{A}$ is symmetric with respect to the x-axis, therefore independent to the ϕ angle 1.8. This is true since we can choose freely the z-axis and y-axis orientation in space around the loop. Then we can choose the point $\mathbf{p}$ to lie on the zx-plane or the yx-plane; in both cases, we will obtain that one of the two $d\mathbf{l}$ components $d\mathbf{l_z}$ and $d\mathbf{l_y}$ will cancel themselves out as we integrate over the loop.

For example, if we consider the point to lie on the yx-plane, then take a point on the loop where $d\mathbf{l}$ is and its symmetric w.r.t. the y-axis, the component $d\mathbf{l_y}$ of the first will cancel itself out with the one of the second.

We can write the length of a circumference as $l = r\,\alpha$, where α is the subtended angle by the length l and r the radius. In addition, we express $\mathbf{e}_\phi$ using the Cartesian basis:

$$\mathbf{e}_\phi = -\sin\phi\,\mathbf{e_y} + \cos\phi\,\mathbf{e_z}$$

Then, $d\mathbf{l}$ magnitude depends on the differential angle $d\phi$ and the radius b, and has the same direction as $\mathbf{e}_\phi$:

$$d\mathbf{l} = b\,d\phi\,\mathbf{e}_\phi = b\,d\phi\,(-\sin\phi\,\mathbf{e_y} + \cos\phi\,\mathbf{e_z}) \tag{2.23}$$

For every $d\mathbf{l}$, there is another symmetrically located differential length element on the other side of the y-axis that will contribute an equal amount to $\mathbf{A}$ in the $\mathbf{e_z}$ direction but will cancel the contribution of $d\mathbf{l}$ in the $\mathbf{e_y}$ direction. Since $\mathbf{e_z} = \mathbf{e}_\phi$, if point P lies on the yx-plane, equation 2.22 can be written as:

$$\mathbf{A} = \mathbf{e}_\phi \frac{\mu I b}{4\pi} \int_0^{2\pi} \frac{\cos\phi}{r_1}\,d\phi \tag{2.24}$$

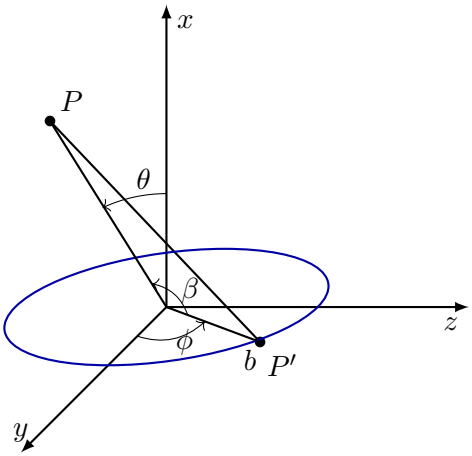

Figure 2.2 Magnetic dipole representation with the OPP' triangle

Computing the Integral in Spherical Coordinates

Firstly, we find r_1 by applying the law of cosines on the triangle OPP', Figure 2.2:

We start with the equation for r_1:

$$r_1^2 = r^2 + b^2 - 2rb\cos\beta$$

Since we are on the xy-plane, we can write the $\mathbf{b}$ and $\mathbf{r}$ vectors as:

$$\mathbf{r} = r\sin\theta\,\mathbf{e_y} + r\cos\theta\,\mathbf{e_x}$$

$$\mathbf{b} = b\cos\phi\,\mathbf{e_y} + b\sin\phi\,\mathbf{e_z}$$

Then we find the angle β between $\mathbf{r}$ and $\mathbf{b}$:

$$\cos\beta = \frac{\mathbf{b}\cdot\mathbf{r}}{rb} = \sin\theta\cos\phi$$

We have obtained a formula for r_1:

$$r_1 = \sqrt{r^2 + b^2 - 2rb\sin\theta\cos\phi}$$

Simplifying further, we get:

$$r_1^2 = r^2\left(1 + \frac{b^2}{r^2} - 2\,\frac{b}{r}\sin\theta\cos\phi\right)$$

Using the same assumption as before, that the loop is very small with respect to r (i.e., $b \ll r$ and therefore $b^2 \ll r^2$), we can write:

$$r_1 \approx r\left(1 - 2\,\frac{b}{r}\,\sin\theta\cos\phi\right)^{1/2}$$

Then we compute the inverse $\frac{1}{r_1}$ and use the Taylor approximation to the first derivative, considering $x = 2\,\frac{b}{r}\,\sin\theta\cos\phi$, which tends to zero, we obtain:

$$\frac{1}{r_1} \approx \frac{1}{r}\left(1 + \frac{b}{r}\sin\theta\cos\phi\right) \tag{2.25}$$

Now, substituting 2.25 in 2.24, we can calculate the integral for $\mathbf{A}$ over the entire loop:

$$\mathbf{A} = \mathbf{e}_\phi \frac{\mu I b}{4\pi}\int_0^{2\pi}\left(1 + \frac{b}{r}\sin\theta\cos\phi\right)\cos\phi\, d\phi$$

Since b, r, and θ do not depend on ϕ, the first integral is zero, and the second one gives π:

(1) $$\int_0^{2\pi}\cos\phi\, d\phi = \sin\phi\Big|_0^{2\pi} = 0$$

(2) $$\int_0^{2\pi}\cos^2\phi\, d\phi = \int_0^{2\pi}\frac{1+\cos(2\phi)}{2}\, d\phi = \frac{1}{2}\cdot 2\pi + \frac{1}{2}\cdot 0 = \pi$$

Therefore, we obtain:

$$\mathbf{A} = \mathbf{e}_\phi \frac{\mu I b^2}{4\, r^2}\sin\theta \tag{2.26}$$

Magnetic Field Intensity H

Finally, we can now obtain an expression in spherical coordinates of the magnetic field intensity $\mathbf{H}$, by first finding $\mathbf{B}$ using 2.5 and then inverting 2.6.
We compute the curl of $\mathbf{A}$ in spherical coordinates 1.20:

$$\mathbf{B} = \nabla\times\mathbf{A} = \frac{1}{r^2\sin\theta}\begin{vmatrix} \mathbf{e_r} & \mathbf{e}_\theta r & \mathbf{e}_\phi r\sin\theta \\ \frac{\partial}{\partial r} & \frac{\partial}{\partial\theta} & \frac{\partial}{\partial\phi} \\ 0 & 0 & r\sin\theta A_\phi \end{vmatrix} =$$

$$= \frac{1}{r^2\sin\theta}\left(\mathbf{e_r}\frac{\partial}{\partial\theta}(r\sin\theta A_\phi) - \mathbf{e}_\theta\frac{\partial}{\partial r}(r\sin\theta A_\phi)\right)$$

where A_ϕ is the magnitude of the vector field found in 2.26, while A_r and A_θ are zero since the potential has only the $\mathbf{e}_\phi$ direction. Substituting 2.26:

$$\mathbf{B} = \frac{1}{r^2\sin\theta}\frac{\mu b^2 I}{4\pi}\left(\mathbf{e_r}\frac{2}{r}\cos\theta\sin\theta + \mathbf{e}_\theta r\sin^2\theta\, r^{-2}\right) =$$
$$= \frac{\mu I b^2}{4\, r^3}\left(\mathbf{e_r}2\cos\theta + \mathbf{e}_\theta\sin\theta\right)$$

Then, by inverting equation 2.6, we obtain the final expression for $\mathbf{H}$:

$$\mathbf{H} = \frac{I b^2}{4\, r^3}\left(\mathbf{e_r}2\cos\theta + \mathbf{e}_\theta\sin\theta\right) \tag{2.27}$$

which can also be expressed using the Cartesian unit vectors $(\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z)$ by substituting $\mathbf{e}_r$ with 1.17 and $\mathbf{e}_\theta$ with 1.18:

$$\mathbf{H} = \frac{Ib^2}{4\pi\, r^3}\left[(2\cos^2\theta - \sin^2\theta)\mathbf{e}_x + 3\cos\theta\sin\theta\cos\phi\,\mathbf{e}_y + 3\cos\theta\sin\theta\sin\phi\,\mathbf{e}_z\right]$$

or in vector form:

$$\mathbf{H} = \frac{Ib^2}{4\,r^3}\begin{bmatrix} 2\cos^2\theta - \sin^2\theta \\ 3\cos\theta\sin\theta\cos\phi \\ 3\cos\theta\sin\theta\sin\phi \end{bmatrix}$$

We can finally find the expression for $\mathbf{H}$ using only Cartesian coordinates by inverting the equations in 1.8.2:

$$\mathbf{H} = \frac{Ib^2}{4\,r^5}\begin{bmatrix} 2x^2 - y^2 - z^2 \\ 3xy \\ 3xz \end{bmatrix} \tag{2.28}$$

This expression has been derived by applying the Pythagorean identity.

2.4 Normalized Source Strength

The magnetic field intensity $\mathbf{H}$ and the magnetic flux density $\mathbf{B}$ are vector fields sensible to the orientation of the source coordinate system. The expression for $\mathbf{H}$, as given in equation 2.28, is expressed with respect to a coordinate system centered at the center of the current loop, as we have seen in the previous section. However, we need to introduce a quantity which is invariant with respect to the orientation of the electromagnetic target.

The magnetic gradient tensor is simply the transposed Jacobian of the magnetic flux density $\mathbf{B}$, and it is defined as follows:

$$\begin{aligned}\mathbf{G} &= \begin{bmatrix} \frac{\partial B_x}{\partial x} & \frac{\partial B_y}{\partial x} & \frac{\partial B_z}{\partial x} \\ \frac{\partial B_x}{\partial y} & \frac{\partial B_y}{\partial y} & \frac{\partial B_z}{\partial y} \\ \frac{\partial B_x}{\partial z} & \frac{\partial B_y}{\partial z} & \frac{\partial B_z}{\partial z} \end{bmatrix} \\ &= \begin{bmatrix} B_{xx} & B_{yx} & B_{zx} \\ B_{xy} & B_{yy} & B_{zy} \\ B_{xz} & B_{yz} & B_{zz} \end{bmatrix}\end{aligned} \tag{2.29}$$

It represents the spatial rate of change of the magnetic field vector $\mathbf{B}$ along the three mutually orthogonal directions of the Cartesian coordinates.

In most cases, rotational invariants, such as the Frobenius norm of $\mathbf{G}$ and any combination of its eigenvalues, are sensitive to the direction of the target magnetic moment vector $\mathbf{m}$ [52]. The magnetic moment represents the strenght and orientation of the magnetic field, and for a magnetic dipole of a current loop it is defined as follows, [23]:

$$\mathbf{m} = I\,\mathbf{S} \tag{2.30}$$

where I is the current flowing in the loop and $\mathbf{S}$, is the normal to the area of the loop and it is not to be confused with S in 2.14, which is the area of the section of the wire. In our representation of the magentic dipole, with the Cartesian axes defined in Figure 2.1, the direction of the magnetic moment is the x-axis, and the magnitude is:

$$m = I\,\pi\,b^2 \tag{2.31}$$

Instead the NSS[1], a tensor invariant calculated from the eigenvalues of the magnetic gradient tensor $\mathbf{G}$, does not depend on the magnetization direction and it is completely isotropic around the magnetic dipole [53]. This follows from the magnetic gradient tensor $\mathbf{G}$ being symmetric and traceless, thanks to Maxwell's equations.

In particular from the fourth equation 2.4, and from the definition of gradient 1.1 we deduce the traceless property:

$$\nabla \cdot \mathbf{B} = 0 \Rightarrow \frac{\partial B_x}{\partial x} + \frac{\partial B_y}{\partial y} + \frac{\partial B_z}{\partial z} = 0 \Rightarrow B_{xx} + B_{yy} + B_{zz} = 0 \tag{2.32}$$

Instead, from the second Maxwell's equation (2.2), in the case of the quasi-static field assumption we used before, and considering regions of space where there is an absence of electric currents ($\mathbf{J} = 0$), the curl of $\mathbf{B}$ is zero ($\nabla \times \mathbf{B} = 0$), [17]. Then, from the definition of curl (1.7), we deduce the symmetry property:

$$\begin{aligned} \nabla \times \mathbf{B} = 0 \Rightarrow \frac{\partial B_x}{\partial y} - \frac{\partial B_y}{\partial x} = 0, \quad \frac{\partial B_x}{\partial z} - \frac{\partial B_z}{\partial x} = 0, \quad \frac{\partial B_y}{\partial z} - \frac{\partial B_z}{\partial y} = 0 \\ \Rightarrow B_{xy} = B_{yx}, \quad B_{xz} = B_{zx}, \quad B_{yz} = B_{zy} \end{aligned} \tag{2.33}$$

The NSS is a combination of the eigenvalues of the tensor $\mathbf{G}$. In order to find the eigenvectors of a matrix, we need to solve the equation:

$$\mathbf{G}\mathbf{v}_i = \lambda_i \mathbf{v}_i \tag{2.34}$$

which is true for all pairs of eigenvalue λ_i and eigenvector $\mathbf{v}_i$. The eigenvalues are found by solving the characteristic polynomial, defined as:

$$\det(\mathbf{G} - \lambda \mathbf{I}) = 0 \tag{2.35}$$

where $\mathbf{I}$ is the identity matrix. Expanding the determinant yields the cubic equation:

$$\lambda^3 - I_1\lambda^2 + I_2\lambda - I_3 = 0 \tag{2.36}$$

where I_1 is the trace of the gradient tensor $\mathbf{G}$, I_2 is the sum of the principal minors, and I_3 is the determinant of $\mathbf{G}$. However, since $\mathbf{G}$ is traceless, meaning that the coefficient of λ^2 vanishes, $I_1 = 0$, we can simplify the cubic equation as done in [17]:

$$\lambda^3 + I_2\lambda - I_3 = 0 \tag{2.37}$$

The invariants I_1, I_2, and I_3 are found following the linear algebra definitions mentioned before and simplified thanks to the symmetric property:

[1] NSS stands for Normalized Source Strength

1. I_1 (Trace):

$$I_1 = B_{xx} + B_{yy} + B_{zz} = 0$$

2. I_2 (Sum of principal minors):

$$\begin{aligned} I_2 &= B_{xx}B_{yy} + B_{yy}B_{zz} + B_{xx}B_{zz} - (B_{xy}B_{yx} + B_{yz}B_{zy} + B_{xz}B_{zx}) = \\ &= B_{xx}B_{yy} + B_{yy}B_{zz} + B_{xx}B_{zz} - (B_{xy}^2 + B_{yz}^2 + B_{xz}^2) \end{aligned}$$

3. I_3 (Determinant):

$$\begin{aligned} I_3 &= B_{xx}(B_{yy}B_{zz} - B_{zy}B_{yz}) - B_{yx}(B_{xy}B_{zz} - B_{zy}B_{xz}) + B_{zx}(B_{xy}B_{yz} - B_{yy}B_{xz}) = \\ &= B_{xx}B_{yy}B_{zz} + 2B_{xy}B_{yz}B_{xz} - B_{xx}B_{yz}^2 - B_{yy}B_{xz}^2 - B_{zz}B_{xy}^2 \end{aligned}$$

It is shown later in 2.4.1 that these coefficients are rotational invariants of the tensor $\mathbf{G}$, meaning that they are unchanged by a rotation of the coordinate axes. They have the neat property that they can be simply expressed directly in terms of the tensor components with respect to any Cartesian reference frame. Each distinct root of the cubic equation defines a corresponding eigenvalue of the tensor. From linear algebra we know that for each eigenvalue λ_i, the associated eigenvectors can be found as non-zero vectors $\mathbf{v}_i$ that satisfy 2.34. Also, since $\mathbf{G}$ is a symmetric real 3×3 matrix, all its eigenvalues are real, and the eigenvectors corresponding to distinct eigenvalues are orthogonal. It is always possible to construct an orthonormal set of the three eigenvectors, even in cases where the eigenvalues are degenerate (i.e., two or more eigenvalues are equal). Furthermore, it is demonstrated later in 2.4.1 that the eigenvalues λ_{min}, λ_{med}, and λ_{max} are rotational invariants of the tensor [17]. Therefore, any combination of the eigenvalues constitutes a rotational invariant and the NSS is defined as:

$$\text{NSS} = \sqrt{\lambda_{\text{med}}^2 - \lambda_{\text{min}}\lambda_{\text{max}}} \tag{2.38}$$

Additionally, it is shown that the NSS is inversely proportional to the fourth power of the distance between the computed point in space and the source [52]:

$$\text{NSS} = \frac{3\mu_0 m}{4\pi r^4} \tag{2.39}$$

From which it is also evident that it is a rotational invariant (since it dipends only on the distance r) and also that it does not depend on the direction of the magnetic moment $\mathbf{m}$, only on its magnitude.

2.4.1 Demonstration of Rotation Invariance

Let's define the gradient operator ∇ in any Cartesian coordinate system (x, y, z), given by

$$\nabla = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{bmatrix} \tag{2.40}$$

as a column vector.
Let (x, y, z) be the Cartesian coordinates in the original frame, and (x', y', z') denote the coordinates in a rotated frame. The rotation is defined by a rotation matrix, which for simplicity, we assume to be a rotation around the z-axis by an angle θ:

$$\mathbf{R}_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The new coordinates (x', y', z') after the rotation are related to the original coordinates by:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \mathbf{R}_z(\theta) \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Using the chain rule, we express the derivatives with respect to the rotated coordinates in terms of the original derivatives:

$$\frac{\partial}{\partial x'} = \cos(\theta)\frac{\partial}{\partial x} - \sin(\theta)\frac{\partial}{\partial y}$$

$$\frac{\partial}{\partial y'} = \sin(\theta)\frac{\partial}{\partial x} + \cos(\theta)\frac{\partial}{\partial y}$$

$$\frac{\partial}{\partial z'} = \frac{\partial}{\partial z}$$

Thus, the gradient operator in the rotated coordinate system can be expressed in matrix form as:

$$\nabla' = \mathbf{R}_z(\theta)\,\nabla \tag{2.41}$$

Now, let $\mathbf{B}$ be a 3D vector field exspressed as a column vector as well:

$$\mathbf{B}(x, y, z) = \begin{bmatrix} B_x(x, y, z) \\ B_y(x, y, z) \\ B_z(x, y, z) \end{bmatrix}$$

The components of the vector field after the rotation are given by:

$$\mathbf{B}' = \mathbf{R}_z(\theta)\,\mathbf{B} \tag{2.42}$$

Then, we can rewrite the definition of the gradient tensor 1.1, in matrix form [1]:

$$\mathbf{G} = \nabla\mathbf{B}^\top$$

Which becomes in the rotated coordinates (x', y', z'):

$$\mathbf{G}' = \nabla'\mathbf{B}'^\top$$

Thus, substituting 2.41 and 2.42 into the expression for G', we obtain:

$$\mathbf{G}' = \mathbf{R}_z(\theta)\, \nabla\, (\mathbf{R}_z(\theta)\, \mathbf{B})^\top$$

Using the linear algebra property of the transpose of the product of two matrices:

$$\mathbf{G}' = \mathbf{R}_z(\theta)\, \nabla\, (\mathbf{B}^\top \mathbf{R}_z^\top(\theta))$$

Recognizing that $\nabla\, \mathbf{B}^\top$ is the original gradient $\mathbf{G}$, we finally obtain the same result as [1]:

$$\mathbf{G}' = \mathbf{R}_z(\theta)\, \mathbf{G}\, \mathbf{R}_z^\top(\theta) \tag{2.43}$$

Firstly, we focus on the eigenvalues of $\mathbf{G}$, namely λ_{min}, λ_{med}, and λ_{max}. To demonstrate rotational invariance, we recall the characteristic polynomial 2.35, which defines the eigenvalues of $\mathbf{G}$. Under a coordinate rotation, the tensor $\mathbf{G}$ transforms as shown in (2.43). Then, the characteristic equation for the rotated tensor is:

$$\det(\mathbf{G}' - \lambda \mathbf{I}) = \det\Big(\mathbf{R}_z(\theta)\, \mathbf{G}\, \mathbf{R}_z^\top(\theta) - \lambda \mathbf{I}\Big)$$

Using Binet's Theorem and the property of orthogonal matrices $\det(\mathbf{R}) = \det(\mathbf{R}^\top) = 1$:

$$\det\left(\mathbf{R}_z(\theta)\, (\mathbf{G} - \lambda \mathbf{I})\, \mathbf{R}_z^\top(\theta)\right) = \det(\mathbf{G} - \lambda \mathbf{I}) \tag{2.44}$$

Consequently, the eigenvalues λ_{min}, λ_{med}, and λ_{max} of the tensor $\mathbf{G}$ remain unchanged, proving that they are rotationally invariant.
Secondly, we now demonstrate how I_2 and I_3 are rotational invariants with respect to a rotation of the coordinates systems. I_2 is defined as the sum of the principal minors of order 2 of $\mathbf{G}$. From linear algebra we know that the sum of principal minors is a linear combination of the eigenvalues of a matrix, which are in turn rotational invariants, as we just demonstrated, therefore I_2 is a rotational invariant as well.
Similarly, the determinant of $\mathbf{G}$, I_3, is also invariant under orthogonal transformations. This is agian due to the fact that the determinat of orthogonal matrices is 1 and thanks to Binet's Theorem:

$$\det(\mathbf{G}') = \det(\mathbf{R}_z(\theta)\, \mathbf{G}\, \mathbf{R}_z^\top(\theta)) = \det(\mathbf{R}_z(\theta))\, \det(\mathbf{G})\, \det(\mathbf{R}_z^\top(\theta)) = \det(\mathbf{G})$$

As I_2 and I_3 are constructed respectively from the sum of principal minors and the determinant of $\mathbf{G}$, both quantities remain invariant under any rotation of the coordinate system.

Chapter 3

Mathematical Model

3.1 Single Victim Case

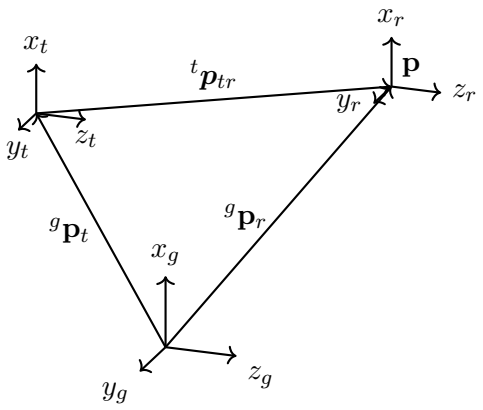

Figure 3.1 Inertial frames in the single victim case

Three Cartesian coordinate frames are defined as [44] and shown in 3.1:

(i) Frame g (global): denoted as $F_g = (O_g, x_g, y_g, z_g)$, is the global (inertial) frame with origin O_g, fixed reference for all objects in space.

(ii) Frame r (receiver ARTVA): denoted as $F_r = (O_r, x_r, y_r, z_r)$, is the body right-hand frame associated with the receiver installed on the drone.

(iii) Frame t (transmitter ARTVA): denoted as $F_t = (O_t, x_t, y_t, z_t)$, is the body right-hand frame associated with the transmitter worn by the victim.

Assumption
We assume that the receiver frame F_r, relative to the ARTVA instrument coincides with the fixed right-hand body frame F_b, for sake of semplicity as in [10].

The position of O_r relative to O_t is indicated by the vector $\mathbf{p}_{tr} \in \mathbb{R}^3$, with $\mathbf{p}_{tr} = \mathbf{p}_r - \mathbf{p}_t$, while the positions of O_r and O_t relative to O_g are indicated, respectively, by the vectors $\mathbf{p}_r \in \mathbb{R}^3$ and $\mathbf{p}_t \in \mathbb{R}^3$. We use the apex g, r or t on the left of the vector to indicate in which frame the vector is expressed , e.g. $^t p$. If it is not specified, we assume the global (inertial)

frame. We denote the *special orthogonal group* of order three as $SO(3)$, which is the set of all 3×3 orthogonal matrices with determinant equal to one:

$$SO(3) = \{\mathbf{R} \in \mathbb{R}^{3\times3} \mid \mathbf{R}^\top\mathbf{R} = \mathbf{I},\, \det(\mathbf{R}) = 1\}$$

Therefore, we indicate with $\mathbf{R}_t^g \in SO(3)$ the rotation matrix of frame F_t with respect to frame F_g, it rotates frame g into frame t and transforms vectors described in t to g, as in [13].
As mentioned in the Introduction 1, the ARTVA instrument can be equipped with three antennas aligned along the receiver frame axes x_r, y_r, and z_r, corresponding to the longitudinal, lateral, and vertical directions of the sensor case, respectively. In both the single victim and multiple victims cases, we assume that the ARTVA receiver position $\mathbf{p}_r$ and orientation $\mathbf{R}_r$ in the inertial frame are known, as in [44].

The electromagnetic field measured by the receiver, defined as ${}^r\mathbf{H}$, is given by the projection of vector field ${}^t\mathbf{H}$ onto the F_r frame, corrupted by some sensor noise:

$${}^r\mathbf{H} = \mathbf{R}_t^r\, {}^t\mathbf{H} + {}^r\mathbf{W}(t) \tag{3.1}$$

where ${}^r\mathbf{W}(t) : \mathbb{R} \to \mathbb{R}^3$ represents the EMI[1] expressed in the receiver frame. Note that the noise is bounded and it is modeled as white noise ${}^r\mathbf{W}(t) \sim \mathcal{N}(0, \sigma^2\mathbf{I})$.

3.1.1 Magnitude of Magnetic Field Intensity H

We have found an expression of $\mathbf{H}$ in spherical coordinates, 2.27, whose magnitude is found as:

$$|\mathbf{H}| = \frac{Ib^2}{4\,r^3}\sqrt{4\cos^2\theta + \sin^2\theta} = \frac{Ib^2}{4\,r^3}\sqrt{3\cos^2\theta + 1} \tag{3.2}$$

Approximation

We use the same approximation in [44] in order to remove the non-linearity given by the square root term $\sqrt{3\cos^2\theta + 1}$. Therefore we approximate:

$$\frac{1}{\sqrt[3]{3\cos^2\theta + 1}} \approx \frac{1}{a^2}\cos^2\theta + \frac{1}{b^2}\sin^2\theta$$

of which the polar plot is shown in Figure 3.2 when a and b have values 1.291 and 1.028, respectively, which minimize the relative mean squared error = 0.123%.

Thus, the square root term becomes:

$$\sqrt{3\cos^2\theta + 1} \approx \frac{1}{\left(\frac{1}{a^2}\cos^2\theta + \frac{1}{b^2}\sin^2\theta\right)^{3/2}} \tag{3.3}$$

Using the approximation (3.3) in (3.2):

$$|\mathbf{H}| = \frac{Ib^2}{4\,r^3}\left(\frac{1}{a^2}\cos^2\theta + \frac{1}{b^2}\sin^2\theta\right)^{2/3} \tag{3.4}$$

[1] EMI stands for electromagnetic interference

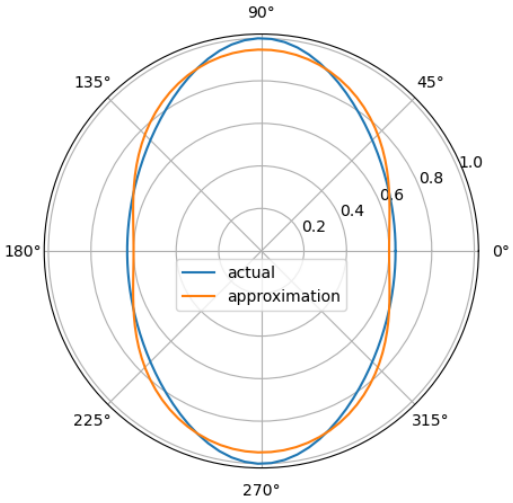

Figure 3.2 Polar plot of the actual function $\sqrt{3\cos^2\theta + 1}$ in blue and the approximated one $\frac{1}{a^2}\cos^2\theta + \frac{1}{b^2}\sin^2\theta$ in orange.

Now we can express the magnitude using the Cartesian coordinates relative to the frame of the transmitter ARTVA F_t. If we consider the point ${}^t\mathbf{p}_{tr}$

$$ {}^t\mathbf{p}_{tr} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} $$

having these coordinates (x, y, z) in frame t, then remembering r from 1.8.1 and $\cos\theta$ from 1.8.2:

$$ \begin{cases} r^2 = x^2 + y^2 + z^2 \\ \cos\theta = \frac{x}{r} \end{cases} $$

Substituting the expressions for r and $\cos\theta$ in 3.4:

$$ |\mathbf{H}| = \frac{Ib^2}{4} \frac{1}{(x^2+y^2+z^2)^3} \left(\frac{1}{a^2} \frac{x^2}{x^2+y^2+z^2} + \frac{1}{b^2} \frac{y^2+z^2}{x^2+y^2+z^2} \right)^{2/3} $$

After simplifications and further calculations, we obtain:

$$ |\mathbf{H}| = \frac{m}{4\pi} \left(\frac{(ab)^2}{b^2x^2 + a^2(y^2+z^2)} \right)^{3/2} \tag{3.5} $$

where we call $I\,\pi\,b^2$ the magnetic moment magnitude m, as seen in 2.31. Then, we can define η as [44]:

$$ \eta = \left(\frac{m}{4\pi\,|\mathbf{H}|} \right)^{2/3} \cdot (ab)^2 = $$

by substituting 3.5:

$$= \left(\frac{m}{4\pi \frac{m}{4\pi} \left(\frac{(ab)^2}{b^2x^2+a^2(y^2+z^2)} \right)^{3/2}} \right)^{2/3} \cdot (ab)^2 =$$
$$= \left(\left(\frac{b^2x^2 + a^2(y^2+z^2)}{(ab)^2} \right)^{3/2} \right)^{2/3} \cdot (ab)^2$$

So:

$$\eta = b^2x^2 + a^2(y^2+z^2) \tag{3.6}$$

3.1.2 Finding the ARTVA position

In order to find the victim's position $\mathbf{p}_t$ with respect to the global frame F_g, we need to use homogeneous transformations [13]. Also from Figure 3.1, we can express the position of the receiver $\mathbf{p}_r$ in the inertial (global) frame as the sum of the other two vectors:

$$^g\mathbf{p}_r = {}^g\mathbf{p}_t + {}^t\mathbf{p}_{tr}$$
$$^t\mathbf{p}_{tr} = \mathbf{R}_g^t \, {}^g\mathbf{p}_{tr}$$

where $\mathbf{R}_g^t$ is the rotation matrix that rotates frame t to g.

From which we can find $\mathbf{p}_t$ by multiplying by ${\mathbf{R}_g^t}^\top$ since $\mathbf{R}_g^t$ is orthogonal (${\mathbf{R}_g^t}^\top = {\mathbf{R}_g^t}^{-1}$):

$$^t\mathbf{p}_{tr} = {\mathbf{R}_g^t}^\top (\mathbf{p}_r - \mathbf{p}_t) \tag{3.7}$$

In addition, remembering how we defined the coordinates of ${}^t\mathbf{p}_{tr}$, then from linear algebra:

$$x = \mathbf{e}_x^\top \, {}^t\mathbf{p}_{tr}$$
$$y = \mathbf{e}_y^\top \, {}^t\mathbf{p}_{tr}$$
$$x = \mathbf{e}_z^\top \, {}^t\mathbf{p}_{tr}$$

also,

$$x^2 = x \cdot x = \left(\mathbf{e}_x^\top \, {}^t\mathbf{p}_{tr} \right)^\top \cdot \left(\mathbf{e}_x^\top \, {}^t\mathbf{p}_{tr} \right)$$

and the same is valid for the other two coordinates, we will omit the calculations for the other two from now on. We can then substitute the expression we found for ${}^t\mathbf{p}_{tr}$ 3.7 and apply linear algebra properties of the transpose:

$$(\mathbf{ABC})^\top = \mathbf{C}^\top \mathbf{B}^\top \mathbf{A}^\top$$

to calculate the transpose of $\mathbf{e}_x^\top {\mathbf{R}_g^t}^\top (\mathbf{p}_r - \mathbf{p}_t)$:

$$(\mathbf{e}_x^\top {\mathbf{R}_g^t}^\top (\mathbf{p}_r - \mathbf{p}_t))^\top = (\mathbf{p}_r - \mathbf{p}_t)^\top \mathbf{R}_g^t \mathbf{e}_x$$

The expression for x^2 then becomes:

$$x^2 = (\mathbf{p}_r - \mathbf{p}_t)^\top \, \mathbf{R}_g^t \mathbf{e}_x \mathbf{e}_x^\top {\mathbf{R}_g^t}^\top \, (\mathbf{p}_r - \mathbf{p}_t) \tag{3.8}$$

Furthermore:

$$\mathbf{e}_x\mathbf{e}_x^\top = \begin{pmatrix}1\\0\\0\end{pmatrix}\begin{pmatrix}1 & 0 & 0\end{pmatrix} = \text{diag}(1,0,0)$$

and:

$$\mathbf{e}_y\mathbf{e}_y^\top = \text{diag}(0,1,0)$$
$$\mathbf{e}_z\mathbf{e}_z^\top = \text{diag}(0,0,1)$$

Lastly we substitute the result found in 3.8 in the expression of η 3.6:

$$\begin{aligned}\eta = \; & b^2\left(\mathbf{p}_r-\mathbf{p}_t\right)^\top \mathbf{R}_g^t\mathbf{e}_x\mathbf{e}_x^\top{\mathbf{R}_g^t}^\top\left(\mathbf{p}_r-\mathbf{p}_t\right)+\\ & + a^2\left(\mathbf{p}_r-\mathbf{p}_t\right)^\top \mathbf{R}_g^t\mathbf{e}_y\mathbf{e}_y^\top{\mathbf{R}_g^t}^\top\left(\mathbf{p}_r-\mathbf{p}_t\right)+\\ & + a^2\left(\mathbf{p}_r-\mathbf{p}_t\right)^\top \mathbf{R}_g^t\mathbf{e}_z\mathbf{e}_z^\top{\mathbf{R}_g^t}^\top\left(\mathbf{p}_r-\mathbf{p}_t\right)=\end{aligned}$$

collect common terms,

$$= \left(\mathbf{p}_r-\mathbf{p}_t\right)^\top \mathbf{R}_g^t\left(b^2\mathbf{e}_x\mathbf{e}_x^\top + a^2\mathbf{e}_y\mathbf{e}_y^\top + a^2\mathbf{e}_z\mathbf{e}_z^\top\right){R_t^g}^\top\left(\mathbf{p}_r-\mathbf{p}_t\right)=$$

$$= \left(\mathbf{p}_r-\mathbf{p}_t\right)^\top \mathbf{R}_g^t\,\text{diag}(b^2,a^2,a^2){\mathbf{R}_g^t}^\top\left(\mathbf{p}_r-\mathbf{p}_t\right) \tag{3.9}$$

We call the $\mathbf{R}_g^t\,\text{diag}(b^2,a^2,a^2){\mathbf{R}_g^t}^\top$ matrix $\mathbf{M}$ and the diagonal matrix $\text{diag}(b^2,a^2,a^2)$ D.

Symmetry of $\mathbf{M}$

Definition
A matrix $\mathbf{M}\in\mathbb{R}^{n\times n}$ is symmetric if and only if $\mathbf{M}=\mathbf{M}^\top$.
Proof
We compute $\mathbf{M}^\top$:

$$\mathbf{M}^\top = \left(\mathbf{R}_g^t\mathbf{D}{\mathbf{R}_g^t}^\top\right)^\top = \left({\mathbf{R}_g^t}^\top\right)^\top\mathbf{D}^\top{\mathbf{R}_g^t}^\top = \mathbf{R}_g^t\mathbf{D}^\top{\mathbf{R}_g^t}^\top$$

Since $\mathbf{D}$ is a diagonal matrix, it is equal to its transpose $\mathbf{D}^\top=\mathbf{D}$:

$$\mathbf{M}^\top = \mathbf{R}_g^tD{\mathbf{R}_g^t}^\top = \mathbf{M}$$

Final expression for η

By applying the distributive property to 3.9 and since M is symmetric:

$$\eta = \mathbf{p}_r^\top\mathbf{M}\mathbf{p}_r - \mathbf{p}_r^\top\mathbf{M}\mathbf{p}_t - \mathbf{p}_t^\top\mathbf{M}\mathbf{p}_r + \mathbf{p}_t^\top\mathbf{M}\mathbf{p}_t \tag{3.10}$$

The vector $\hat{\mathbf{p}}_t$ gives an estimate of the true position $\mathbf{p}_t$:

$$\hat{\mathbf{p}}_t = \mathbf{M}\mathbf{p}_t$$

and since M is symmetric:

$$\hat{\mathbf{p}}_t^\top = \mathbf{p}_t^\top\mathbf{M}^\top = \mathbf{p}_t^\top\mathbf{M}$$

We substitute these expressions in 3.10 and use the definition of the scalar product:

$$\eta = \mathbf{p}_r^\top \mathbf{M}\mathbf{p}_r - \mathbf{p}_r^\top \hat{\mathbf{p}}_t - \hat{\mathbf{p}}_t^\top \mathbf{p}_r + \mathbf{p}_t^\top \mathbf{M}\mathbf{p}_r = \mathbf{p}_r^\top \mathbf{M}\mathbf{p}_r - 2\mathbf{p}_r^\top \hat{\mathbf{p}}_t + \mathbf{p}_t^\top \mathbf{M}\mathbf{p}_t$$

If we define the coordinates of $\mathbf{p}_r$:

$$\mathbf{p}_r = \begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix}$$

and

$$\mathbf{M} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{12} & m_{22} & m_{23} \\ m_{13} & m_{23} & m_{33} \end{pmatrix}$$

we can compute $\mathbf{p}_r^\top \mathbf{M}\mathbf{p}_r$:

$$\mathbf{p}_r^\top \mathbf{M}\mathbf{p}_r = m_{11}\, x_r^2 \;+\; 2\, m_{12}\, x_r\, y_r \;+\; 2\, m_{13}\, z_r\, x_r \;+\; m_{22}\, y_r^2 \;+\; 2\, m_{23}\, y_r\, z_r \;+\; m_{33}\, z_r^2$$

Then, we obtain a final expression for η as in [44]:

$$\begin{aligned} \eta = {} & m_{11}\, x_r^2 + 2\, m_{12}\, x_r\, y_r + 2\, m_{13}\, z_r\, x_r \\ & + m_{22}\, y_r^2 + 2\, m_{23}\, y_r\, z_r + m_{33}\, z_r^2 \\ & - 2\, x_r\, x_t - 2\, y_r\, y_t - 2\, z_r\, z_t \\ & + \mathbf{p}_t^\top \mathbf{M}\mathbf{p}_t \end{aligned} \tag{3.11}$$

We can rewrite the expression of eta in vector form by defining a vector $\mathbf{\Phi}$, which is a funcion of known variables:

$$\mathbf{\Phi}(\mathbf{p}_r) = \begin{pmatrix} x_r^2 \\ 2x_r y_r \\ 2x_r z_r \\ y_r^2 \\ 2y_r z_r \\ z_r^2 \\ -2x_r \\ -2y_r \\ -2z_r \\ 1 \end{pmatrix} \tag{3.12}$$

and a vector of unknown constants $\mathbf{x}$:

$$\mathbf{x}(\mathbf{p}_t) = \begin{pmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{22} \\ m_{23} \\ m_{33} \\ \mathbf{p}_t \\ \rho \end{pmatrix} \tag{3.13}$$

where $\rho := \mathbf{p}_t^\top \mathbf{M} \mathbf{p}_t$.
Then η exspressed in a compact vector form is:

$$\eta = \mathbf{\Phi}^\top(\mathbf{p}_r)\, \mathbf{x}(\mathbf{p}_t) \tag{3.14}$$

Afterward, we leverage the η signal (function of the ARTVA output of Eq. 3.1) to estimate the the transmitter's position $\mathbf{p}_t$, which corresponds to the victim's location. We omit the estimation of the transmitter orientation $\mathbf{R}_g^t$, since it is not critical in either the single-victim or multiple-victim cases.

Final Model

In the final model of the single-victim case, we consider one victim to which it is attached an ARTVA transmitter and $m \in \mathbb{N}$ receivers, each rigidly attached to each drone. We define the position of the j-th drone at sample time τ_k with $\mathbf{p}_{r_j}(\tau_k) \in \mathbb{R}^3$ where $k, i \in \mathbb{N}$ and $j = 1, \ldots, m$.
As described in [10] and as seen in the previous sections, we model the intensity of the ARTVA signal using η in Eq. 3.14 as an output function of the form $y : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$:

$$y(\mathbf{p}_{r_j}, \tau_k) = \mathbf{\Phi}^\top(\mathbf{p}_{r_j})\, \mathbf{x}(\mathbf{p}_t) + w_t(\mathbf{p}_{r_j}, \mathbf{p}_t, \tau_k),$$

where $\mathbf{\Phi}(\mathbf{p}_{r_j})$ is the vector of known variables relative to each k-th receiver and and $w : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ represents the measurement noise and model mismatch introduced by the approximation. Note that w is realted to the EMI defined in 3.1, as demonstrated in [10].

3.1.3 Recursive Least Square

We propose a decentralized estimation algorithm to address our task of localizing avalanche victims. A decentralized approach offers several advantages over a centralized one, particularly in scenarios characterized by limited computational resources and large-scale deployments. By distributing computation across multiple agents (the UAVs), this method reduces computational overload, enhancing scalability as the number of agents increases.

Furthermore, decentralization provides improved robustness to "communication noise" by removing dependence on a central computation unit. This autonomy enables the system to adapt to varying communication conditions, such as intermittent information availability or delays, thus supporting continued operation in challenging environments such asmountain regions and adverse weather conditions, like in our scenario. While decentralized methods may exhibit slower convergence and slightly higher estimation errors due to partial information exchange, they remain effective and efficient in situations where centralized coordination is impractical or demands excessive resources.

To estimate the constant vector $\mathbf{p}_t$, we assume the position of each UAV, $\mathbf{p}_{r_j}(\tau_k)$, as known. Each UAV in the network collects data independently and transmits information only to its neighboring agents. Each UAV then employs its own instance of the estimation algorithm, populating the necessary matrices and vectors in the equations to generate an independent estimate of the victim's position:

$$\mathbf{Y}(\tau_k) = \mathbf{H}(\tau_k)\mathbf{x}(\mathbf{p}_t) + \mathbf{W}(\tau_k)$$

where

$$\mathbf{Y}(\tau_k) = \text{col}(y(\mathbf{p}_{r_1}(\tau_k), \tau_k), \ldots, y(\mathbf{p}_{r_m}(\tau_k), \tau_k)),$$
$$\mathbf{H}(\tau_k) = \text{col}(\mathbf{\Phi}^\top(\mathbf{p}_{r_1}(\tau_k)), \ldots, \mathbf{\Phi}^\top(\mathbf{p}_{r_m}(\tau_k))),$$
$$\mathbf{W}(\tau_k) = \text{col}(w(\mathbf{p}_{r_1}(\tau_k), \mathbf{p}_t, \tau_k), \ldots, v(\mathbf{p}_{r_m}(\tau_k), \mathbf{p}_t, \tau_k)).$$

The model above enables estimation of $\mathbf{x}$ using the affine term $\mathbf{H}(\tau_k)$ through the RLS [2] algorithm:

$$\begin{cases} \hat{\mathbf{x}}(\tau_{k+1}) = \hat{\mathbf{x}}(\tau_k) + \mathbf{S}^{-1}(\tau_k)\mathbf{H}(\tau_k)\left(\mathbf{Y}(\tau_k) - \mathbf{H}(\tau_k)^\top \hat{\mathbf{x}}(\tau_k)\right) \\ \mathbf{S}(\tau_{k+1}) = \beta\mathbf{S}(\tau_k) + \mathbf{H}(\tau_k)\mathbf{H}(\tau_k)^\top \end{cases}$$

where $\beta \in (0, 1)$ is a forgetting factor, and $\mathbf{S}(\tau_0) = \mathbf{S}_0 \in \mathbb{R}^{3\times3}$.
The transmitter's position is estimated by

$$\hat{\mathbf{p}}_t(\tau_k) = \mathbf{x}^{-1}(\hat{\mathbf{x}}(\tau_k)) \tag{3.15}$$

where $\mathbf{x}^{-1} : \mathbb{R}^{10} \to \mathbb{R}^3$ is well-defined as shown in [10].
Note that, in both the RLS method applied for the single-victim case and in the multiple-victim algorithm, we assume the use of a precise positioning system, such as an ideal GPS.

3.1.4 Average Consensus Filter

A PI-ACF[3] was employed to compute a decentralized moving average of the UAVs' estimates. The ACF dynamics equations are described below:

$$\begin{cases} \dot{z}^l = \gamma(\alpha^j - z^j) - K_P \sum_{l\in\mathcal{N}_j}(z^j - z^l) + K_I \sum_{l\in\mathcal{N}_j}(\omega^j - \omega^j) \\ \dot{\omega}^j = -K_I \sum_{l\in\mathcal{N}_j}(z^j - z^l) \end{cases}$$

where:

- z^j: moving average computed by the k-th UAV;
- ω^j: integral variable, representing cumulative error with neighbors;
- α^j: time-varying set-point, mean of neighboring UAVs' estimates;
- $\mathcal{N}_j$: set of neighbors of the k-th UAV;
- γ: hyperparameter adjusting convergence rate;
- K_P and K_I: proportional and integral control gains.

This consensus mechanism enables UAVs to converge to a shared moving average, as each minimizes differences with its neighbors. The tracking term $\gamma(\alpha^j - z^j)$ guides the j-th UAV towards its set-point, while the proportional control term $-K_P \sum_{j\in\mathcal{N}_j}(z^j - z^l)$ ensures alignment with its neighbors. The integral control term $K_I \sum_{j\in\mathcal{N}_j}(\omega^j - \omega^l)$ adds cumulative adjustment based on historical differences, addressing steady-state errors.

This algorithm supports convergence of the UAVs to a common moving average, even with limited communication. It allows the fleet to track changing set-points while reducing differences with neighbors in order to reach consensus.

[2] RLS stands for Recursive Least Squares
[3] PI-ACF stands for Proportional Integral Average Consensus Filter

3.1.5 Normalized Source Strength

In both the single-victim and multiple-victims cases, we can find the the NSS not only numerically but also analytically, in this section we will detail how we compute the NSS for a single source with both methods.

Analitycal method

Since we have an expression of the magnetic field intensity ${}^t\mathbf{H}$ in Cartesian coordinates (in the transmitter reference frame) 2.28, we can compute the derivatives and therefore the gradient tensor ${}^t\mathbf{G}$ analytically as:

$$ {}^t\mathbf{G} = \frac{1}{r^{5/2}} \begin{bmatrix} 3x(-2x^2+3y^2+3z^2) & 3y(-4x^2+y^2+z^2) & 3z(-4x^2+y^2+z^2) \\ 3y(-4x^2+y^2+z^2) & 3x(x^2-4y^2+z^2) & -15xyz \\ 3z(-4x^2+y^2+z^2) & -15xyz & 3x(x^2+y^2-4z^2) \end{bmatrix} \tag{3.16} $$

where r is of course the distance, $r = \sqrt{x^2+y^2+z^2}$.

Note once again the symmetry and traceless property of the gradient tensor $\mathbf{G}$ is evident, when we compute it explicitly. Also note that for simplicity we computed the gradient using $\mathbf{H}$ and not $\mathbf{B}$, since they vary only by the constant permeability μ. We also omitted the constants I, b, since they don't depend on the coordinates (x, y, z) neither, and therefore do not influence the results.

In Figure 3.3, we show the gradients for a single source centered at the origin of the space, when the global frame F_g coincides with the trasmitter (source) frame F_t and also when there is no rotation between these frames and the receiver frame F_r of the drones. In this way, we have that the magnetic field intensity ${}^t\mathbf{H} = {}^g\mathbf{H}$ can be computed using the inertial coordinates, and also the NSS at any point in space coincides with the signal received by any drone located at that point. The gradients are calculated on a grid of equally spaced points in the range $[-5, 5]$, when we fix the z coordinates at 3 m, considering a reasonable flying height for the drones. We need to fix one coordinate in order to be able to plot a function of 3 variables, but also because we assume the drones flying always at a fixed height, as it will be explained more in depth lately.

After having found the gradient tensor $\mathbf{G}$, we numerically compute its eigenvalues and we find the NSS using the definition 2.38. In Figure 3.4, we plot the NSS distribution over the same grid, with the same z value fixed. These values will be the signals received by the drones at their location in space in the multiple victims case, which will be explained more in depth in the following paragraph.

Instead, let's consider the case when the source location is translated with respect to the global frame F_g and the transmitter frame F_t is rotated with respect to the same one by a rotation $\mathbf{R}_g^t$, like in the previous section. Furthermore, we also consider the rotation matrix $\mathbf{R}_t^r$ of the transmitter frame F_t with respect to the receiver frame F_r. Then, we compute the coordinates of any point in space with respect to the transmitter frame as in 3.7. Using these coordinates we find the magnetic intensity vector ${}^t\mathbf{H}$, expressed in the trasmitter frame and compute the gradient tensor ${}^t\mathbf{G}$. The magnetic field read by the receiver, denoted by ${}^r\mathbf{H}$, is given by the projection of vector ${}^t\mathbf{H}$ onto the F_r frame, as in Eq.3.1 but without considering the noise for the moment:

$$ {}^r\mathbf{H} = \mathbf{R}_t^r \, {}^t\mathbf{H} \tag{3.17} $$

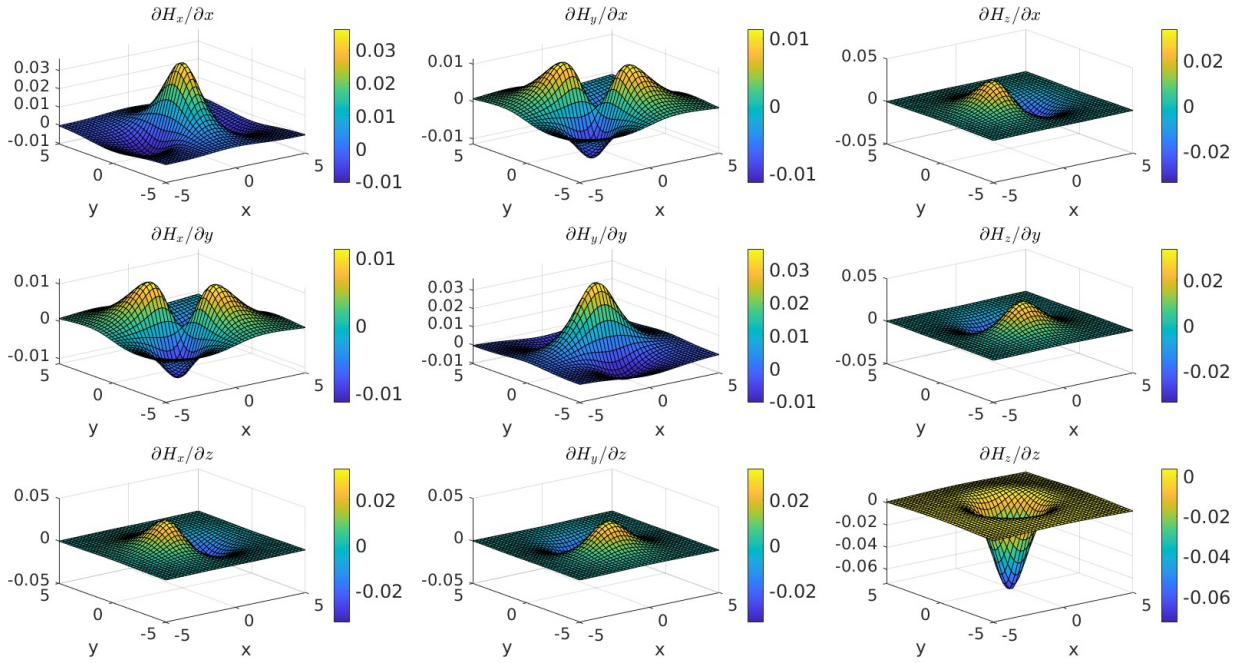

Figure 3.3 Plot of the gradients $\frac{\partial H_i}{\partial q_j}$, where $i, j \in \{x, y, z\}$, H_i are the scalar functions relative to the i-th component of the magnetic field intensity $\mathbf{H}$ and q_j are the Cartesian coordinates. The gradients are the components H_{ij} of the gradient tensor $\mathbf{G}$ when computed analytically as in Eq. 3.16. This example considers a single source located at the center $(0, 0, 0)$ of the space, with no rotations between the coordinate frames F_g, F_r, and F_t.

Therefore, we apply Eq. 2.43 to find the gradient tensor in the receiver frame, which represents the signal obtained in a real world scenario:

$$ {}^{r}\mathbf{G} = \mathbf{R}_t^r \, {}^{t}\mathbf{G} \, \mathbf{R}_t^{r\top} \tag{3.18} $$

In Figure 3.5, we see how the gradients plots vary with respect to the gradients in Figure 3.3, since the rotations introduce naturally a modification in the elettromagnetic field. However, the rotated gradient tensor matrix ${}^{r}\mathbf{G}$ remains symmetric. In addition, in Figure 3.6, it is also evident how the NSS distribution remains invariant under all the different rotations, and it is just affected by the translation in the sense that the peak now is located at the new location of the source.

Central difference method

We also simulate and compare another method to find the gradient tensor matrix ${}^{r}\mathbf{G}$, which comes closer to a real world scenario. We call this method numerical and it consists of obtaining the signal measured by the instrument, which is the electromagnetic field intensity ${}^{r}\mathbf{H}$, and then compute directly its relative gradient tensor using a numerical approximation. We simulate and simplify the way the authors of [52] use a cubic structure measurement array composed of eight TAMs[4] in order to obtain the magnetic field information and then

[4]TAM stands for Tri-Axial Magnetometer

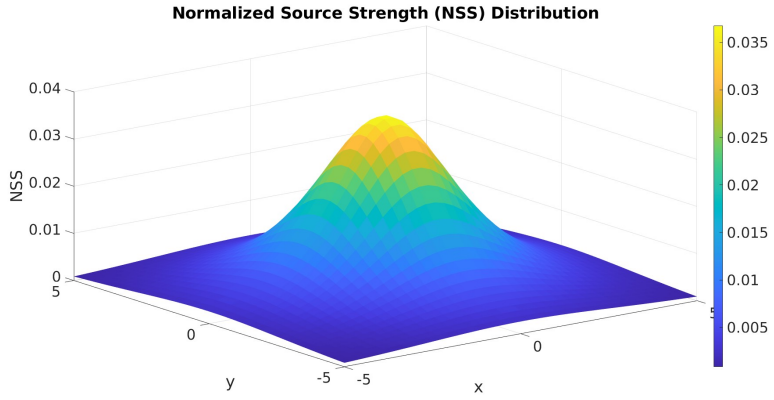

Figure 3.4 Plot of the NSS values computed at each point on the grid, which represent the signal received by the drones, in the case of a single source located at the center $(0, 0, 0)$ of the space when there are no rotations between the coordinates frames F_g, F_r, F_t..

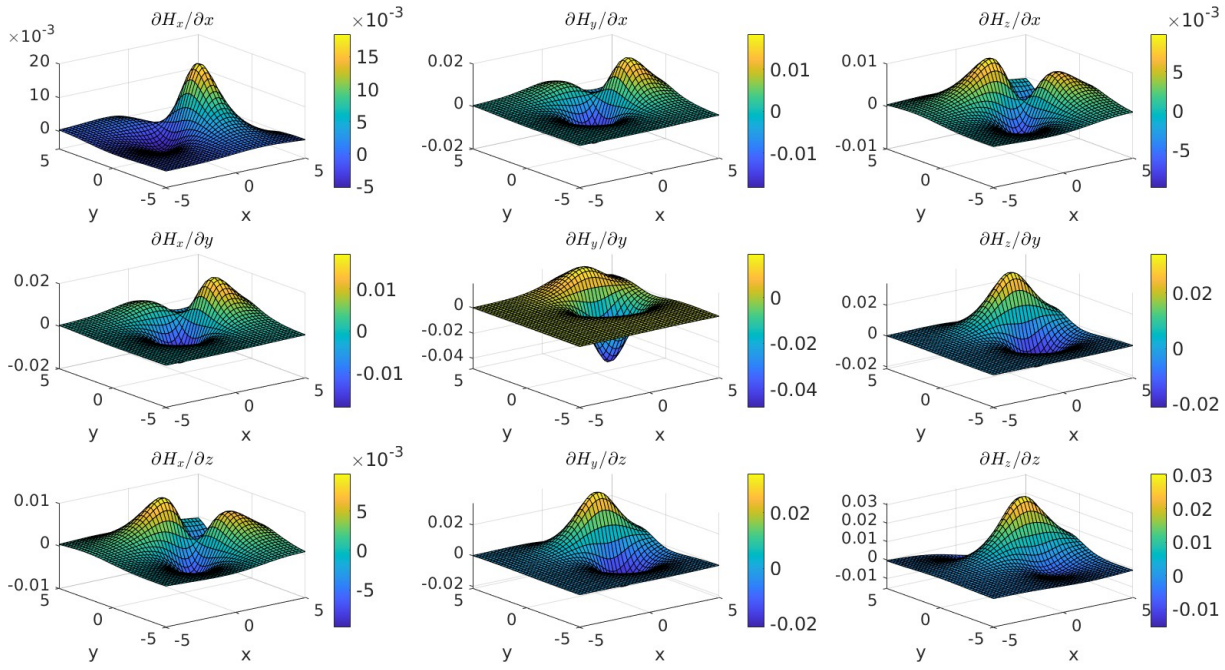

Figure 3.5 Plot of the gradient tensor $\mathbf{G}$ elements when computed analytically after the position of the signle source is translated to $(2, 2, 0)$ and an introduction of the different rotations, expressed using $\mathbf{R}_t^r$ and $\mathbf{R}_t^g$.

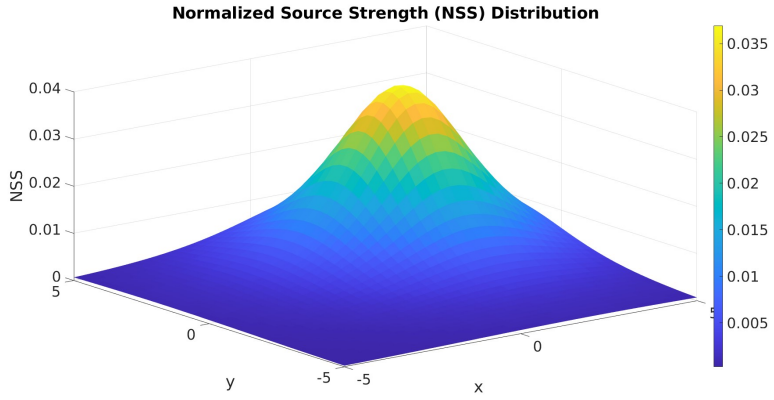

Figure 3.6 Plot of the NSS computed analytically after the position of the signle source is translated to $(2, 2, 0)$ and different roations are introduced, expressed using $\mathbf{R}_t^r$ and $\mathbf{R}_t^g$.

compute the gradient tensor ${}^r\mathbf{G}$.

In order to numerically compute the gradient of a 3D vector field $\mathbf{H}(\mathbf{p}) : \mathbb{R}^3 \to \mathbb{R}^3$, we apply the central difference method approximation. Note that we omit the superscript associated with the transmitter frame F_t, as we are referring to a general vector field in Cartesian space. In our scenario, however, we have $\mathbf{p} = {}^t\mathbf{p}_{tr}$, whose coordinates are (x, y, z). As seen before, the gradient matrix $\mathbf{G}$ is a 3×3 matrix and its (i, j)th entry is $G_{ij} = \partial H_i / \partial q_j$, therefore remembering the definition of gradient as column vector 2.40, we can write the gradient tensor as:

$$\mathbf{G} = \begin{bmatrix} \frac{\partial \mathbf{H}}{\partial x} \\ \frac{\partial \mathbf{H}}{\partial y} \\ \frac{\partial \mathbf{H}}{\partial z} \end{bmatrix} = \begin{bmatrix} \nabla H_x & \nabla H_y & \nabla H_z \end{bmatrix} \tag{3.19}$$

Let's consider a small perturbation δ, which perturbs the magnetic field intensity $\mathbf{H}$ in every coordinate direction, for all the components H_i of the vector field. Then, the partial derivative of $\mathbf{H}$ with respect to a coordinate q_j is given by:

$$\frac{\partial \mathbf{H}}{\partial q_j}(\mathbf{p}) \approx \frac{\mathbf{H}(\mathbf{p} + \delta \mathbf{e}_j) - \mathbf{H}(\mathbf{p} - \delta \mathbf{e}_j)}{2\delta} \tag{3.20}$$

where $\mathbf{e}_j$ denotes the unit vector in the j-th direction. The gradient matrix, $\mathbf{G}$, is constructed by repeating this process for each coordinate direction. The j-th row of $\mathbf{G}$ is:

$$\mathbf{G}_j = \frac{\partial \mathbf{H}}{\partial q_j}(\mathbf{p}) = \nabla^\top H_i(\mathbf{p}) \tag{3.21}$$

In Figure 3.7, we show the results obtained by computing the gradient tensor $\mathbf{G}$ numerically, compared to the analytical method, for the case where there are no rotations between reference frames, and the source is located at the origin. We computed the difference between the gradient tensor matrices in the analytical and numerical cases, $\Delta\mathbf{G} = \mathbf{G}_{\text{analytical}} - \mathbf{G}_{\text{numerical}}$, and calculated the Frobenius norm 3.22 of the difference matrix at all points $\mathbf{p}$ in space. The error was found to be on the order of 10^{-13}, indicating a very accurate approximation. The

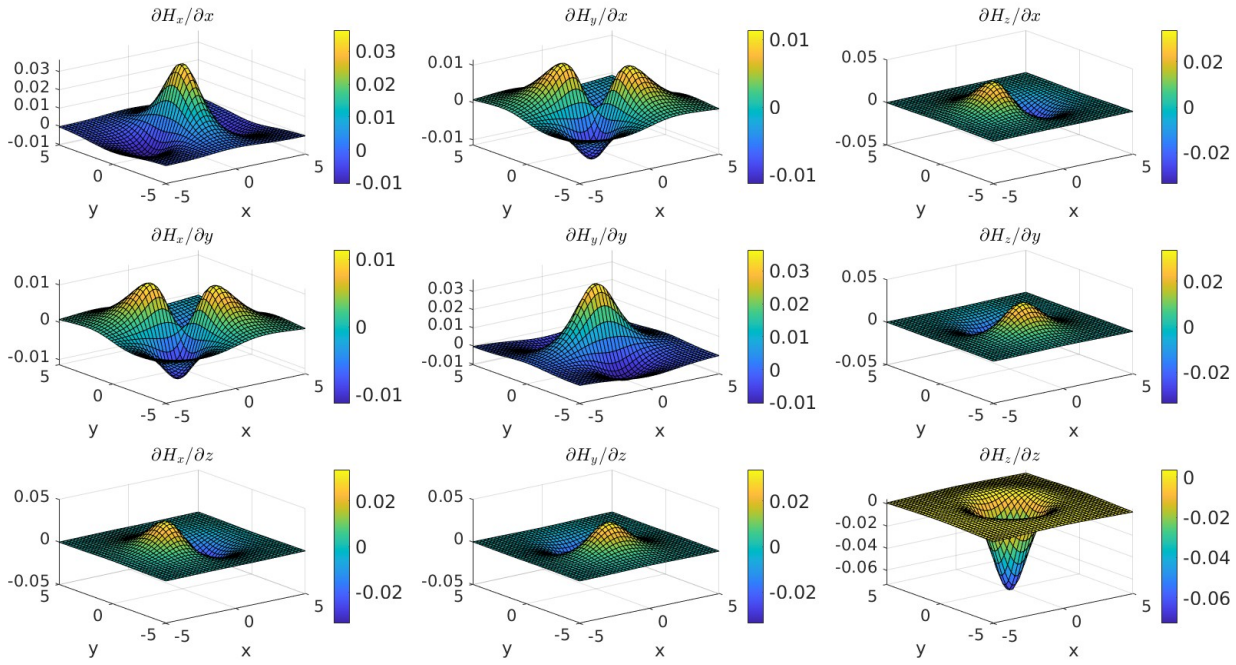

Figure 3.7 Plot of the gradient tensor $\mathbf{G}$ elements when computed numerically as in (3.20), for a single source located at the center $(0, 0, 0)$ of the space, with no rotations between the coordinate frames F_g, F_r, and F_t.

Frobenius norm is given by the square root of the sum of the squared elements δH_{ij} of the difference matrix $\Delta\mathbf{G}$:

$$\|\Delta\mathbf{G}\|_F = \sqrt{\sum_{i=1}^{3}\sum_{j=1}^{3}|\Delta H_{ij}|^2} \tag{3.22}$$

For conciseness, we omit the plot of the NSS since it remains the same, as the gradients did not change.

Next, we need to verify that the gradient tensor matrix remains symmetric under rototranslations of the coordinate frames and that the NSS value also remains invariant. In particular, we apply the following transformations to each point $\mathbf{p}$ in space: firstly we express $\mathbf{p}$ in the transmitter frame using 3.7, which we denoted as ${}^t\mathbf{p}_{tr}$. Then we apply rotation $\mathbf{R}_t^r$ to obtain the point expressed in the receiver coordinates:

$${}^r\mathbf{p}_{tr} = \mathbf{R}_t^r\,{}^t\mathbf{p}_{tr} \tag{3.23}$$

Now, we perturb ${}^r\mathbf{p}_{tr}$ by δ positively and negatively, in the receiver frame:

$$\begin{cases} {}^r\mathbf{p}_{tr} + \delta\mathbf{e}_j \\ {}^r\mathbf{p}_{tr} - \delta\mathbf{e}_j \end{cases} \quad \text{where } j \in \{x_r, y_r, z_r\} \text{ relative to the coordinate frame } F_r.$$

We find the perturbed magnetic field intensity in the transmitter frame using Eq. 2.28: ${}^t\mathbf{H}(\mathbf{R}_t^{r\top}({}^r\mathbf{p}_{tr} \pm \delta\mathbf{e}_j))$. Therefore, we are now able to find the perturbed magnetic field intensity in the receiver frame ${}^r\mathbf{H}$ and apply the central difference approximation 3.20 to compute the gradient tensor in the receiver reference frame ${}^r\mathbf{G}$:

$$\frac{\partial\mathbf{H}}{\partial q_j} = \frac{\mathbf{R}_t^r\,{}^t\mathbf{H}(\mathbf{R}_t^{r\top}({}^r\mathbf{p}_{tr} + \delta\mathbf{e}_j)) - \mathbf{R}_t^r\,{}^t\mathbf{H}(\mathbf{R}_t^{r\top}({}^r\mathbf{p}_{tr} - \delta\mathbf{e}_j))}{2\delta} \tag{3.24}$$

where $j \in \{x_r, y_r, z_r\}$ as before.
Finally, we obtained the rows of ${}^r\mathbf{G}$ and using Eq.3.21, we obtain the complete gradient matrix. Then we can compute the eigenvalues and the NSS with Eq. 2.38.

In Figure 3.8, we see that we obtain the same gradient tensor matrix at all points in space like in the analytical approach, and therefore we verify that it respects the symmetry property. Naturally, the computed NSS remains unchanged as well, as we show in Fig. 3.9.

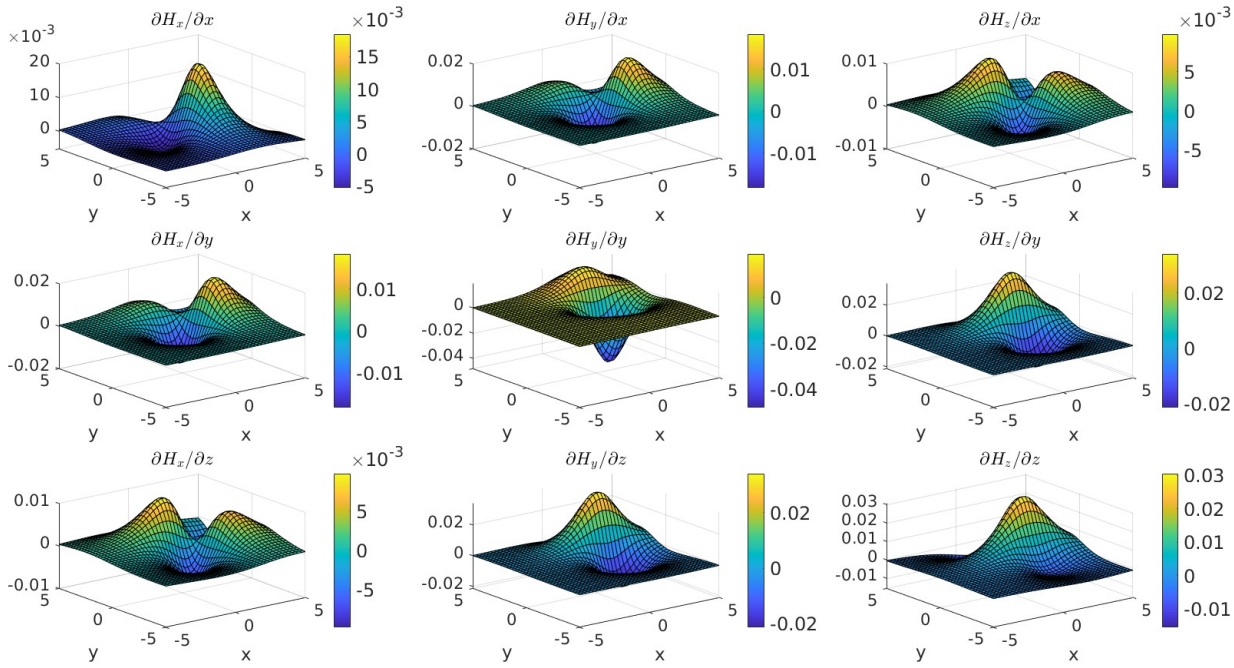

Figure 3.8 Plot of the gradient tensor ${}^{r}\mathbf{G}$ elements when computed numerically after the position of the single source is translated to $(2, 2, 0)$ and an introduction of the different rotations, expressed using $\mathbf{R}_t^r$ and $\mathbf{R}_g^t$.

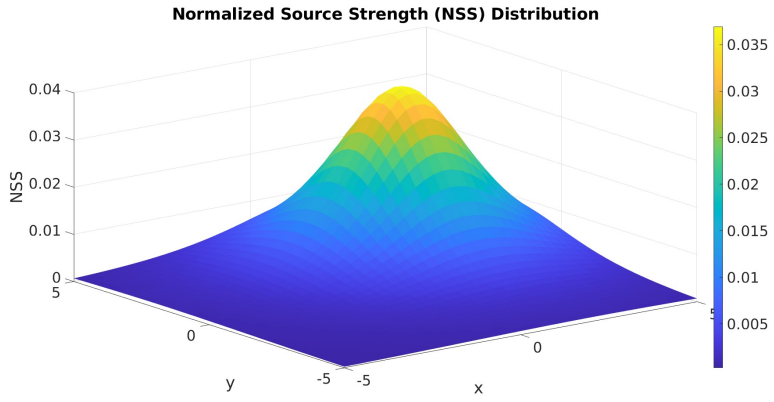

Figure 3.9 Plot of the NSS computed numerically, after the position of the signle source is translated to $(2, 2, 0)$ and different roations are introduced, expressed using $\mathbf{R}_t^r$ and $\mathbf{R}_g^t$.

3.2 Multiple Victims Case

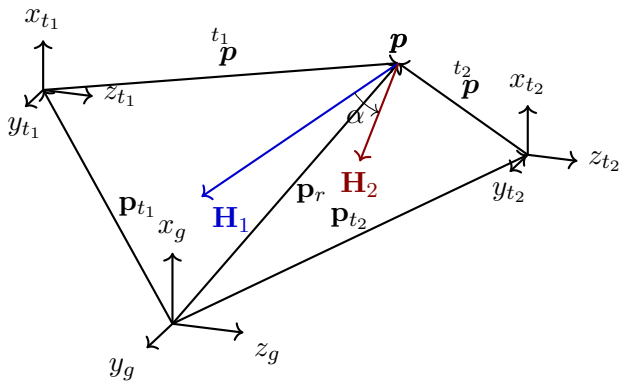

Figure 3.10 Only 2 victims case

In the case of multiple victims, an ARTVA transmitter is attached to each avalanche victim and emits an electromagnetic field. Consequently, the signal received by each drone (receiver) is a superposition of the effects of the multiple electromagnetic sources. In addition, each source has a uniquely oriented body frame, which we will denote with F_{t_i}, where $i = 1, 2, \ldots, n$ and $n \in \mathbb{N}$ is the total number of sources. In addition, we also define the global (inertial) frame F_g and the receiver frame relative to the each drone F_r, in the same way as the single victim case. Then, the relative position of the receiver with respect to each trasmitter ${}^{t_i}\mathbf{p}_{t_i r}$ is denoted by ${}^{t_i}\mathbf{p}$, for semplicity; and the position of the i-th source in the global frame is denoted by $\mathbf{p}_{t_i}$. The relative orientation of frames $F_{t_1}, F_{t_2}, \ldots, F_{t_n}$ with respect to the reference frame F_g are denoted with $\mathbf{R}^g_{t_i} \in SO(3)$. Instead the rotation matrix of the i-th transmitter frame F_{t_i} with respect to the receiver frame F_r is written as $\mathbf{R}^r_{t_i} \in SO(3)$.

Then, considering the receiver drone positioned at point $\mathbf{p}$ in space, the superposition of all the electromagnetic fields can be expressed as the sum of the magnetic field intensity vectors $\mathbf{H}_n$, when projected in the receiver frame F_r, as in [37]:

$$ {}^{r}\mathbf{H}_{\text{tot}} = {}^{r}\mathbf{H}_1 + {}^{r}\mathbf{H}_2 + \cdots + {}^{r}\mathbf{H}_n \tag{3.25} $$

Therefore, since the receiver is rotated with respect to each transmitter, we can rewrite it as:

$$ {}^{r}\mathbf{H}_{\text{tot}} = \mathbf{R}^r_{t_1}\, {}^{t_1}\mathbf{H}_1({}^{t_1}\mathbf{p}) + \mathbf{R}^r_{t_2}\, {}^{t_2}\mathbf{H}_2({}^{t_2}\mathbf{p}) + \cdots + \mathbf{R}^r_{t_n}\, {}^{t_n}\mathbf{H}_n({}^{t_n}\mathbf{p}) \tag{3.26} $$

Remembering Eq.2.27, we can express each single magnetic field intensity vector generated by the i-th magnetic source as:

$$ {}^{t_i}\mathbf{H}_i = \frac{Ib^2}{4\pi r^3} \left(\mathbf{e}_{r_i}\, 2\cos\theta + \mathbf{e}_{\theta_i}\, \sin\theta\right) $$

Furthermore, we use the same homogeneous transformation as in the single victims case, Eq.3.7:

$$ \mathbf{p}_{t_i} + \mathbf{R}^g_{t_i}\, {}^{t_i}\mathbf{p} = \mathbf{p}_r \tag{3.27} $$

which is again inverted:

$$ {}^{t_i}\mathbf{p} = \mathbf{R}^{g\,\top}_{t_i} \left(\mathbf{p}_r - \mathbf{p}_{t_i}\right) \tag{3.28} $$

In Figure 3.10, for clarity, we represent all the frames described before, in the case of only two victims.

3.2.1 Total Magnitude

In the multiple victims case, the use of the total magnitude of the magnetic field intensity $||\mathbf{H}||$ as a mean to infer the position of the multiple sources, is an unfeasible strategy. In this section, we motivate the rationale behind our reasoning, by obtaining the mathematical formula of the total magnitude. Specifically, we aim to demonstrate that, even after the use of the previous approximation 3.3, the total magnitude is highly non-linear and non-invertible (not even partially bijective).
In both the single-victim and multiple ones scenarios, the orientation of the receiver frame relative to the transmitter frame did not affect the magnitude of the magnetic field intensity. This is because the magnitude of any vector field (having the same properties of any vector quantity) is rotationally invariant. Supposing only one source, the projection of the electromagnetic field intensity on the receiver frame is ${}^{r}\mathbf{H}$, and it is related to ${}^{t}\mathbf{H}$ in Eq.3.17; then using simple linear algebra definitions we can write:

$$\|{}^{r}\mathbf{H}\|^2 = \|\mathbf{R}_t^r\,{}^{t}\mathbf{H}\|^2 = (\mathbf{R}_t^r\,{}^{t}\mathbf{H})\cdot(\mathbf{R}_r^t\,{}^{t}\mathbf{H}) = {}^{t}\mathbf{H}^{\top}\,\mathbf{R}_t^{r\top}\,\mathbf{R}_t^r\,{}^{t}\mathbf{H} = {}^{t}\mathbf{H}^{\top}\,{}^{t}\mathbf{H} = \|{}^{t}\mathbf{H}\|^2 \tag{3.29}$$

This shows that the magnitude of the vector field expressed in the receiver coordinate frame F_r is equal to the one expressed in the trasmitter frame F_t.
In the multiple vicims case, the receiver frame is oriented in a different way for each one of the n trasmitter ones. However, the total magnitude is still unaffected by the different rotations, thanks to the result we have just obtained from Eq. 3.29. Let's suppose, for simplicity, the case of only two sources like in Figure 3.10. We want to find the total magnitude of the magnetic field intensity in Eq. 3.26. Therefore, we apply the law of cosines from linear algebra:

$$||\mathbf{H}_{\text{tot}}||^2 = ||{}^{r}\mathbf{H}_1||^2 + ||{}^{r}\mathbf{H}_2||^2 + 2\,||{}^{r}\mathbf{H}_1||\,||{}^{r}\mathbf{H}_2||\cos\alpha$$

where α is the angle between the two vectors $\mathbf{H}_1$ and $\mathbf{H}_2$ on the only plane which contains both vector fields. To use the approximation in Eq. 3.3, we need to express the vector fields in the transmitter reference frame:

$$\begin{aligned}\|\mathbf{H}_{\text{tot}}\|^2 &= \|\mathbf{R}_{t_1}^r\,{}_1^t\mathbf{H}_1\|^2 + \|\mathbf{R}_{t_2}^r\,{}_2^t\mathbf{H}_2\|^2 + 2\,\|\mathbf{R}_{t_1}^r\,{}_1^t\mathbf{H}_1\|\,\|\mathbf{R}_{t_2}^r\,{}_2^t\mathbf{H}_2\|\cos\alpha \\ &= \|{}^{t}\mathbf{H}_1\|^2 + \|{}^{t}\mathbf{H}_2\|^2 + 2\,\|{}^{t}\mathbf{H}_1\|\,\|{}^{t}\mathbf{H}_2\|\cos\alpha.\end{aligned}$$

Substituting Eq.3.5 of the magnetic field magnitude after the approximation, with the Cartesian coordinates of reference frame F_t:

$$\begin{aligned}||\mathbf{H}_{\text{tot}}||^2 = &\left(\frac{m}{4\pi}\right)^2\left(\frac{(ab)^2}{b^2\,x_1^2 + a^2\,(y_1^2+z_1^2)}\right)^3 + \left(\frac{m}{4\pi}\right)^2\left(\frac{(ab)^2}{b^2\,x_2^2 + a^2\,(y_2^2+z_2^2)}\right)^3 + \\ &+2\,\cos\alpha\left(\frac{m}{4\pi}\right)^2\left(\frac{(ab)^2}{b^2\,x_1^2 + a^2\,(y_1^2+z_1^2)}\right)^{3/2}\left(\frac{(ab)^2}{b^2\,x_2^2 + a^2\,(y_2^2+z_2^2)}\right)^{3/2} = \\ = &\left(\frac{m\,(ab)^3}{4\pi}\right)^2\left(\left(\frac{1}{b^2\,x_1^2 + a^2\,(y_1^2+z_1^2)}\right)^3 + \left(\frac{1}{b^2\,x_2^2 + a^2\,(y_2^2+z_2^2)}\right)^3 + \right. \\ &\left. +2\,\cos\alpha\left(\frac{1}{(b^2\,x_1^2 + a^2\,(y_1^2+z_1^2))(b^2\,x_2^2 + a^2\,(y_2^2+z_2^2))}\right)^{3/2}\right)\end{aligned}$$

Note that a and b, have the same values for all the fields since they have been chosen in order to optimize the magnitude of the general magnetic field intensity $\mathbf{H}$. Also we suppose that

the radius b and the current I are the same for all the devices (so same magnetic moment m). Bringing the common terms to the left side and inverting all the fractions:

$$\left(\frac{m\,(ab)^3}{4\pi\,||\mathbf{H}_{\text{tot}}||}\right)^2 = \left(b^2\,x_1^2 + a^2\,(y_1^2 + z_1^2)\right)^3 + \left(b^2\,x_2^2 + a^2\,(y_2^2 + z_2^2)\right)^3 + \\ + 2\,\cos\alpha\left((b^2\,x_1^2 + a^2\,(y_1^2 + z_1^2))(b^2\,x_2^2 + a^2\,(y_2^2 + z_2^2))\right)^{3/2}$$

From this point it is not possible to carry out the mathematical operations that have been carried out in the single-victim case. Therefore, it is not possible to find a mapping as in 3.15, as the above equation is not even partially bijective. Furthermore, the non-invertibility and non linearities, determined by the cross terms, would only increase in the cases of more than two victims. In the following section, we will describe an alternative approach, which employs the NSS as a mean to estimate the position of multiple avalanche victims.

3.2.2 Normalized Source Strength

As already mentioned, the properties of rotational invariance (see Eq.2.44) and rapid decaying of the NSS (see Eq.2.39) are particularly beneficial in the multiple-victims case. In this section, we will describe how we compute the gradient tensor matrix ${}^rG_{\text{tot}}$ and the NSS in the multiple-victims case, for a drone located at point p in space with a rigidly attatched body fram F_t. We employ these procedures to simulate the results over a spatial range of $[10, 10]$ meters, when we fix the z-coordinate fixed at 3 m (i.e., the xy-plane at $z = 3$), in which three victims are positioned at $(0, 0, 0)$, $(-6, -6, 0)$, and $(7, 7, 0)$ respectively.

Analitycal method

The analytical method in the multiple-victims case is similar to the single-victims one; we will describe the general procedure when different orientations are present between the different reference frames F_g, F_r, and F_{t_i}.

Firstly, for each different i-th source we express the point at which the NSS needs to be obtained in the transmitter reference frame F_{t_i}, in the same way as Eq.3.27.

Secondly, we use the analytical expression in Eq.3.16, to compute the gradient tensor in the transmitter frame ${}^{t_i}\mathbf{G}_i$, of the i-th source. Then we transform ${}^{t_i}\mathbf{G}_i$ in ${}^r\mathbf{G}_i$, the gradient expressed in the receiver frame using Eq.2.43.

Finally, we find ${}^r\mathbf{G}_{\text{tot}}$, the total gradient tensor by summing all the gradient tensors relative to the different suorces in the common receiver frame:

$$\begin{aligned} {}^r\mathbf{G}_{\text{tot}} &= \mathbf{R}^r_{t_1}\,{}^{t_1}\mathbf{G}_1({}^{t_1}\mathbf{p})\,\mathbf{R}^{r\,\top}_{t_1} + \cdots + \mathbf{R}^r_{t_n}\,{}^{t_n}\mathbf{G}_n({}^{t_n}\mathbf{p})\,\mathbf{R}^{r\,\top}_{t_n} = \\ &= \mathbf{R}^r_{t_1}\,{}^{t_1}\mathbf{G}_1(R^{g\,\top}_{t_1}(\mathbf{p}_r - \mathbf{p}_{t_1}))\,\mathbf{R}^{r\,\top}_{t_1} + \cdots + \mathbf{R}^r_{t_n}\,{}^{t_n}\mathbf{G}_n(R^{g\,\top}_{t_1}(\mathbf{p}_r - \mathbf{p}_{t_1}))\mathbf{R}^{r\,\top}_{t_n} \end{aligned} \tag{3.30}$$

Note that we sum the gradient matrices, rather than summing the magnetic field intensity vectors ${}^r\mathbf{H}_i$, as the gradient operation is linear. Thanks to linearity, the gradient matrix of the sum of the vectors is equal to the sum of the gradient tensors.
By looking at Fig.3.11 and Fig.3.13 we can observe that the components of the gradient matrix vary in one case with respect to another; however the matrix mantains the symmetry property. Instead, by comparing Fig.3.12 with Fig.3.14 we verify that the NSS distribution is rotationally invariant, like in the single victims case.

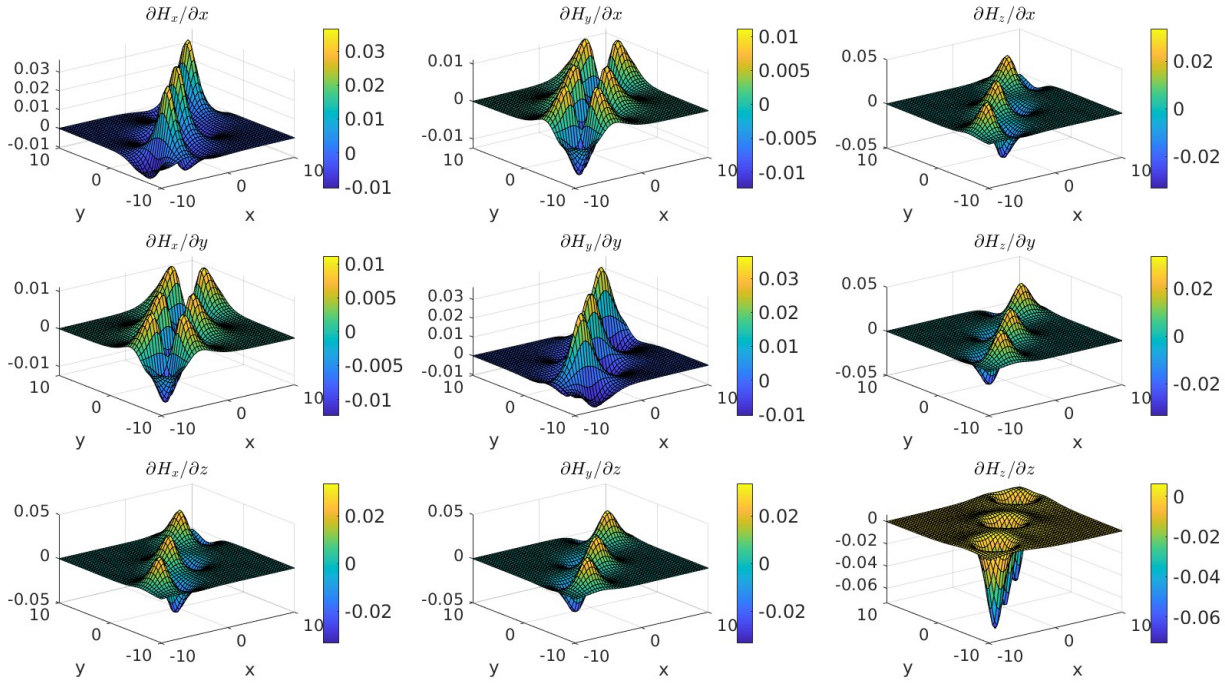

Figure 3.11 Plot of the gradient tensor ${}^{r}\mathbf{G}_{\text{tot}}$ elements when computed analytically in the case of multiple sources positioned at $(0, 0, 0)$, $(-6, -6, 0)$, and $(7, 7, 0)$; with no rotations between the coordinate frames F_g, F_r, and F_t.

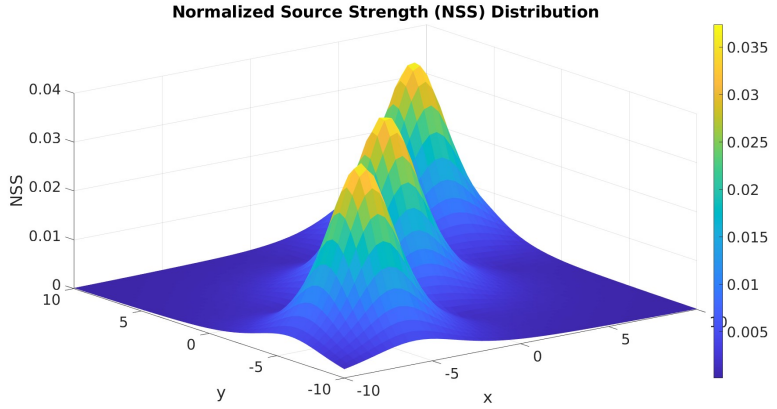

Figure 3.12 Plot of the NSS computed analytically in the case of multiple sources positioned at $(0, 0, 0)$, $(-6, -6, 0)$, and $(7, 7, 0)$; with no rotations between the coordinate frames F_g, F_r, and F_t.

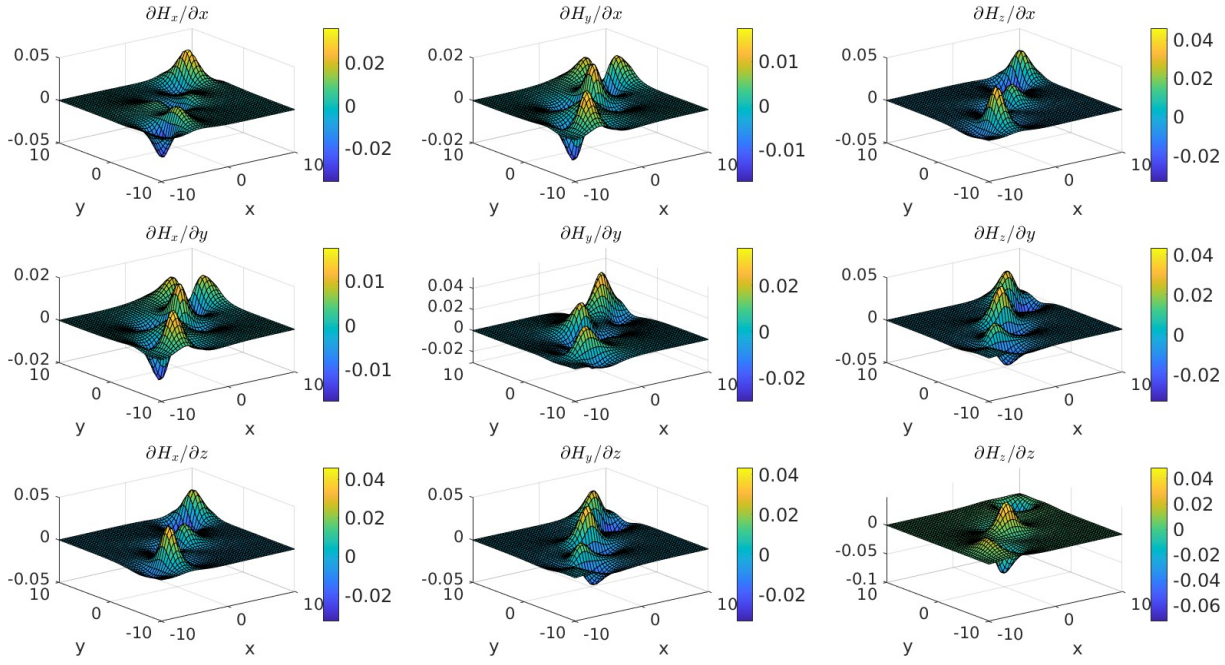

Figure 3.13 Plot of the gradient tensor ${}^{r}\mathbf{G}_{\text{tot}}$ elements when computed analytically in the case of multiple sources positioned at $(0, 0, 0)$, $(-6, -6, 0)$, and $(7, 7, 0)$; with rotations $\mathbf{R}_t^r$ and $\mathbf{R}_g^t$.

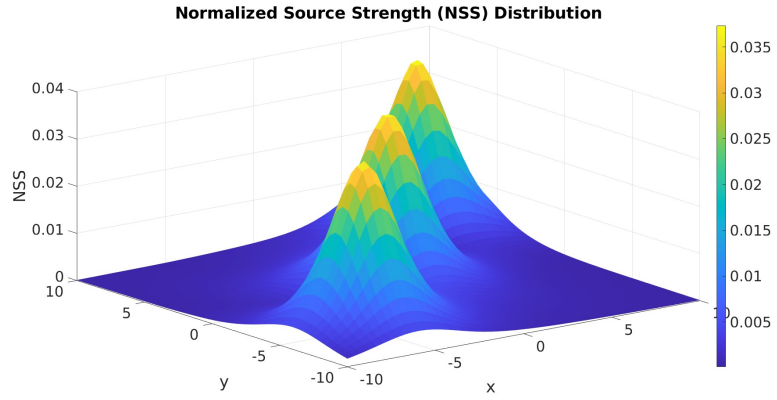

Figure 3.14 Plot of the NSS computed analytically in the case of multiple sources positioned at $(0, 0, 0)$, $(-6, -6, 0)$, and $(7, 7, 0)$; with rotations $\mathbf{R}_t^r$ and $\mathbf{R}_g^t$.

Numerical method

The numerical method in the multiple-victims case is similar to the single-victims one as well; we will describe the general procedure when different orientations are present between the different reference frames F_g, F_r, and F_{t_i}.

We procede in the same way as in the analytical method: for each different i-th source we express the point at which the k-th UAV is localized, in the transmitter reference frame F_t, Eq.3.27. Then we apply rotation $\mathbf{R}_{t_i}^r$ to obtain the point expressed in the receiver coordinates, which we denote by ${}^r\mathbf{p}_{t_i r}$, similarly to Eq.3.23. So that we can perturb ${}^r\mathbf{p}_{t_i r}$ by δ, along each direction j of the receiver coordinates (x_r, y_r, z_r):

$$\begin{cases} {}^r\mathbf{p}_{t_i r} + \delta\mathbf{e}_j \\ {}^r\mathbf{p}_{t_i r} - \delta\mathbf{e}_j \end{cases} \qquad \text{where } j \in \{x_r, y_r, z_r\} \text{ relative to the coordinate frame } F_r.$$

Then, to obtain the perturbed magnetic field intensity and so use Eq.2.28, we need to revert the last transformation and therefore we transpose the rotation matrix $\mathbf{R}_{t_i}^r$ to obtain the reverse transformation:

$$\begin{cases} {}^{t_i}\mathbf{p}^+ = \mathbf{R}_{t_i}^{r\,\top}\,({}^r\mathbf{p}_{t_i r} + \delta\mathbf{e}_j) \\ {}^{t_i}\mathbf{p}^- = \mathbf{R}_{t_i}^{r\,\top}\,({}^r\mathbf{p}_{t_i r} - \delta\mathbf{e}_j) \end{cases}$$

We now have computed the perturbed magnetic field intensity vector in the transmitter frame generated by each one of the i-th source, denoted as ${}^{t_i}\mathbf{H}_i$. Then, we rotatate each perturbed magnetic field vector to express it in the common receiver reference frame, so that we can add the contribution of each electromagnetic field together and obtain ${}^r\mathbf{H}_{\text{tot}}$, which is exactly what is described in Eq.3.26:

$$\begin{cases} {}^r\mathbf{H}_{\text{tot}}^+ = R_r^{t_1}\,{}^r\mathbf{H}_1({}^{t_1}\mathbf{p}^+) + \cdots + R_r^{t_n}\,{}^{t_n}\mathbf{H}_n({}^{t_n}\mathbf{p}^+) \\ {}^r\mathbf{H}_{\text{tot}}^- = R_r^{t_1}\,{}^r\mathbf{H}_1({}^{t_1}\mathbf{p}^-) + \cdots + R_r^{t_n}\,{}^{t_n}\mathbf{H}_n({}^{t_n}\mathbf{p}^-) \end{cases}$$

Finally, we can find the perturbed magnetic field intensity in the receiver frame ${}^r\mathbf{H}_i$ and apply the central difference approximation 3.20 to compute the total gradient tensor in the receiver reference frame ${}^r\mathbf{G}_{\text{tot}}$:

$$\frac{\partial \mathbf{H}_{\text{tot}}}{\partial q_j} = \frac{{}^r\mathbf{H}_{\text{tot}}^+({}^r\mathbf{p}_{t_1 r}, \ldots, {}^r\mathbf{p}_{t_n r}) - {}^r\mathbf{H}_{\text{tot}}^-({}^r\mathbf{p}_{t_1 r}, \ldots, {}^r\mathbf{p}_{t_n r})}{2\delta} \tag{3.31}$$

Finally, we obtained the rows of ${}^r\mathbf{G}_{\text{tot}}$ and using Eq.3.21, we obtain the complete gradient matrix. Then, we can compute the eigenvalues and the NSS with Eq.2.38, in the same way as the single-victim case.

By looking at Fig.3.15 and Fig.3.17 we can observe again that the gradients vary in the case in which the reference frames F_{t_i}, F_r and F_g have different orientations with respect to one another; however the matrix mantains the symmetry property. Instead, by comparing Fig.3.16 with Fig.3.18 we verify that the NSS distribution is rotationally invariant, like in all the other cases both with single or multiple victims.

3.2.3 Final Model

In the final model of the multiple-victims scenario, we consider an ARTVA transmitter attached to each i-th victim among the $n \in \mathbb{N}$ victims, where $i = 1, \ldots, n$. In addition, we

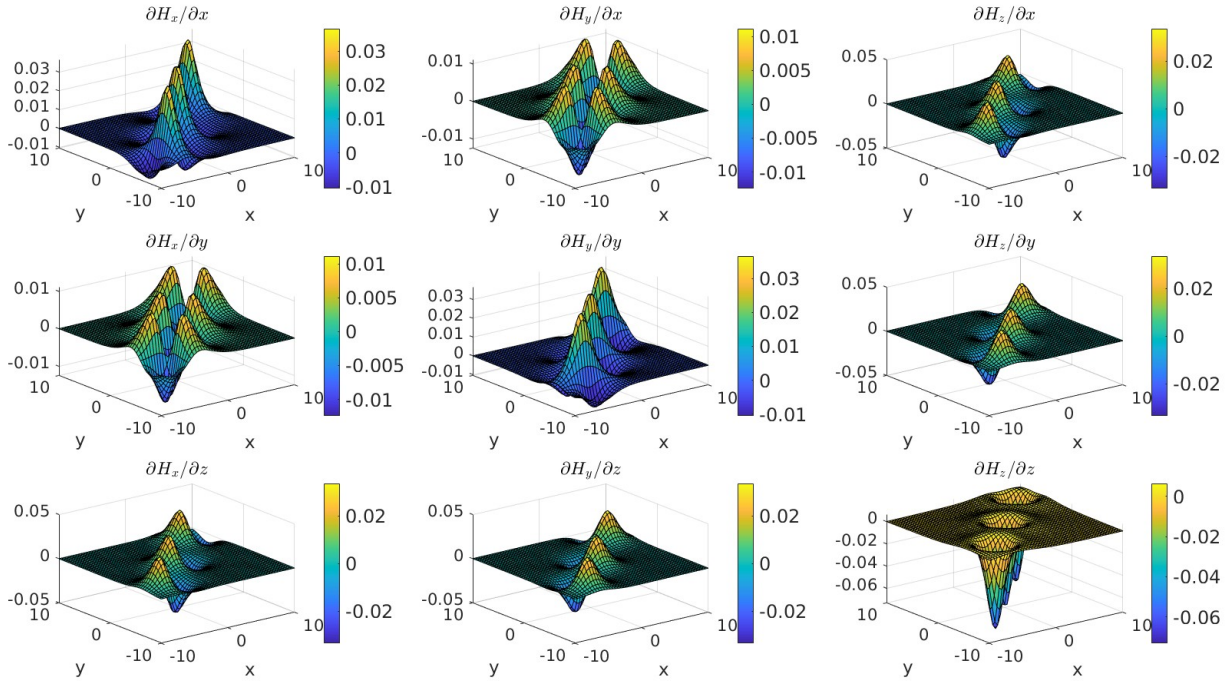

Figure 3.15 Plot of the gradient tensor ${}^{r}\mathbf{G}_{\text{tot}}$ elements when computed numerically in the case of multiple sources positioned at $(0, 0, 0)$, $(-6, -6, 0)$, and $(7, 7, 0)$; with no rotations between the coordinate frames F_g, F_r, and F_t.

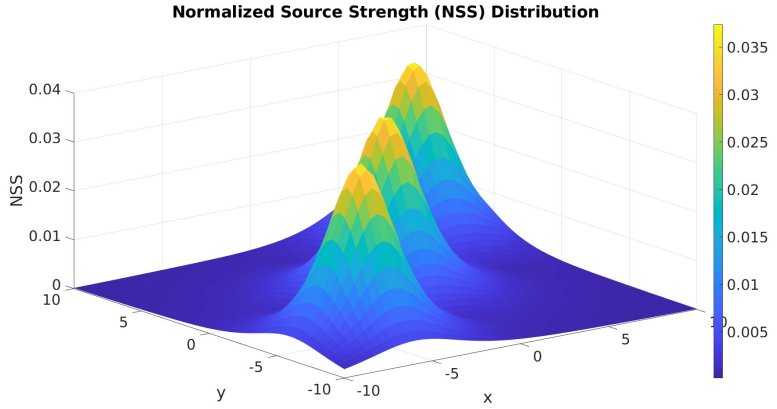

Figure 3.16 Plot of the NSS computed numerically in the case of multiple sources positioned at $(0, 0, 0)$, $(-6, -6, 0)$, and $(7, 7, 0)$; with no rotations between the coordinate frames F_g, F_r, and F_t.

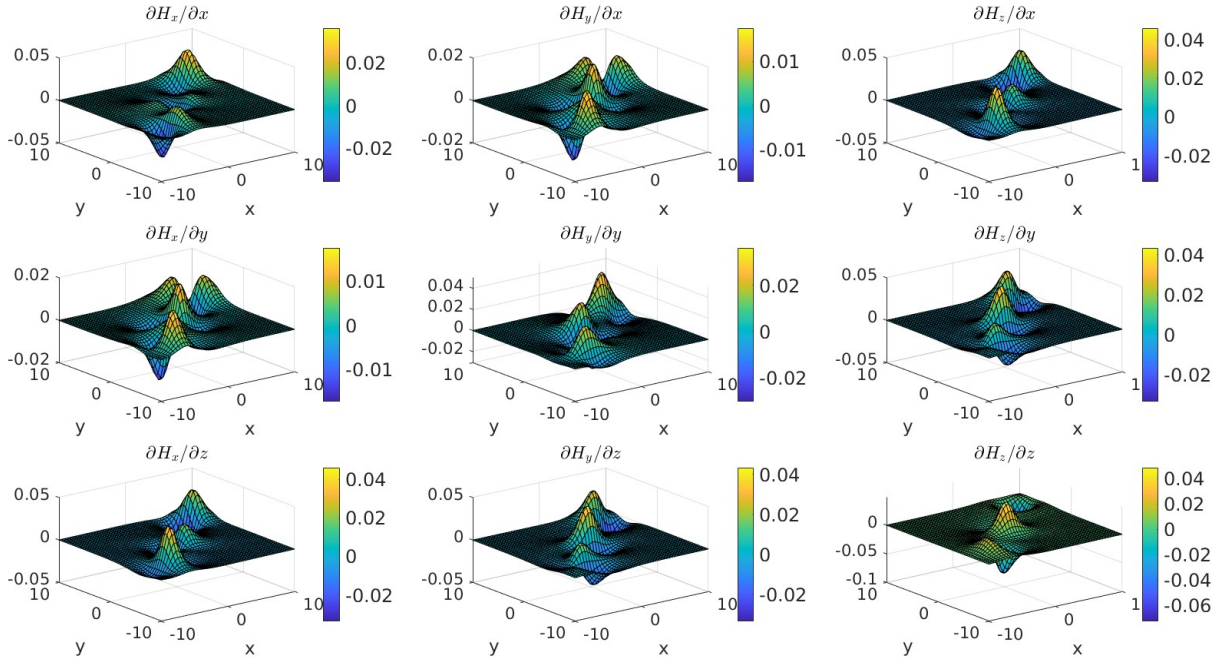

Figure 3.17 Plot of the gradient tensor ${}^{r}\mathbf{G}_{\text{tot}}$ elements when computed numerically in the case of multiple sources positioned at $(0, 0, 0)$, $(-6, -6, 0)$, and $(7, 7, 0)$; with rotations $\mathbf{R}_t^r$ and $\mathbf{R}_g^t$.

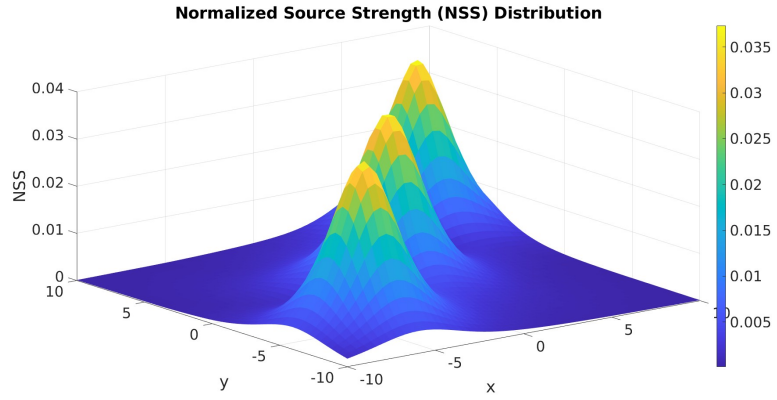

Figure 3.18 Plot of the NSS computed numerically in the case of multiple sources positioned at $(0, 0, 0)$, $(-6, -6, 0)$, and $(7, 7, 0)$; with rotations $\mathbf{R}_t^r$ and $\mathbf{R}_g^t$.

employ a swarm of $m \in \mathbb{N}$ UAVs to locate the position of at least m victims if $m \min n$, otherwise to localize them all. We assume the position of the j-th drone, $\mathbf{p}_{r_j}(\tau_k) \in \mathbb{R}^3$, to be known at every sample time τ_k, where $k, j \in \mathbb{N}$ and $j = 1, \ldots, m$. We also consider each UAV to be equipped, with an ARTVA receiver, or any other instrument capable of detecting the electromagnetic field at different points, like the TAMs in [52]. Since ARTVAs are the most commonly used instruments we compute the NSS using the analytical method, but we note that the numerical method could be employed in the precence of any other suitable equipment.

Knowing the position of the j-th drone $\mathbf{p}_r(\tau_k)$, we compute the NSS using the analytical expression of the gradient tensor, which depends only on the position of the receiver drone in the transmitter frame ${}^{t_i}\mathbf{p}(\tau_k)$. From Eq. 3.27 we express the position of each j-th drone in the relative trasmitter frame F_{t_i} of each i-th source:

$$ {}^{t_i}\mathbf{p}(\tau_k) = {R_{t_i}^g}^\top \left(\mathbf{p}_r(\tau_k) - \mathbf{p}_{t_i}(\tau_k)\right) \tag{3.32} $$

Then, we compute the gradient tensor matrix relative to a single source ${}^{t_i}\mathbf{G}_i({}^{t_i}\mathbf{p})$ using the analytical expression in 3.16:

$$ {}^{t_i}\mathbf{G}_i = \frac{1}{r_i^{5/2}} \begin{bmatrix} 3x_i(-2x_i^2 + 3y_i^2 + 3z_i^2) & 3y_i(-4x_i^2 + y_i^2 + z_i^2) & 3z_i(-4x_i^2 + y_i^2 + z_i^2) \\ 3y_i(-4x_i^2 + y_i^2 + z_i^2) & 3x_i(x_i^2 - 4y_i^2 + z_i^2) & -15x_i y_i z_i \\ 3z_i(-4x_i^2 + y_i^2 + z_i^2) & -15x_i y_i z_i & 3x_i(x_i^2 + y_i^2 - 4z_i^2) \end{bmatrix} \tag{3.33} $$

where r_i is the distance between the i-th source and the receiver position $\mathbf{p}_r(\tau_k)$ and (x_i, y_i, z_i) are the coordinates relative to ${}^{t_i}\mathbf{p}(\tau_k)$.

Subsequently, we sum all the gradient tensors rotated in the common receiver frame F_r to obtain ${}^{r}\mathbf{G}_{\text{tot}}({}^{t_1}\mathbf{p}, \ldots {}^{t_n}\mathbf{p})$, in the same way as Eq. 3.30.

Finally, we conpute the eigenvalues of total gradient tensor to find the NSS relative to all sources using Eq.2.38, at each time step τ_k:

$$ \text{NSS}_{\text{tot}}(\tau_k) = \sqrt{{\lambda_{\text{tot}_2}}^2 - \lambda_{\text{tot}_1}\lambda_{\text{tot}_3}} \tag{3.34} $$

where λ_{tot_1}, λ_{tot_2} and λ_{tot_3} are the ordered (from min to max) eigenvalues relative to the total magnetic gradient tensor.

Therefore, we model the intensity of the signal received by each UAV using the NSS as an output funcion of the form $y_j : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$:

$$ y_j(\mathbf{p}_r, \mathbf{p}_{t_i}, \tau_k) = \mathbf{\Phi}(\mathbf{p}_r, \mathbf{p}_{t_i}, \tau_k) + w_t(\mathbf{p}_r, \mathbf{p}_{t_i}, \tau_k) \tag{3.35} $$

where $\mathbf{\Phi}(\mathbf{p}_r, \mathbf{p}_{t_i}, \tau_k)$ is the NSS relative to each j-th receiver and and $w : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ represents the measurement noise related to the electromagnetic interference, defined in 3.1.

3.2.4 Modified Particle Swarm Optimization with Exclusion Zones

The PSO algorithm is an intelligent population-based stochastic optimization technique that simulates the behavior of a natural biological swarm. As already mentioned, each “particle” represents a potential solution in the search space and it updates its velocity and position based on the historical best positions of both the individual and the entire population.

We asssume that m seeker drones act as m particles moving in the search-space, like in [31]; therefore we denote the position of each particle as the j-th UAV $\mathbf{p}_{r_j}(\tau_k) \in \mathbb{R}^2$, at the τ_k time step.

Assumption

We make the assumption that the UAVs fly at a constant height; therefore we can simplify the search task on the 2D plane. Furthermore, in the standard PSO algorithm [3], the particles are randomly initialized within the search-space. In contrast, our model assumes that the UAVs start from the center of the space, they then move in the search space uniformly and once they reach the desired position the PSO algorithm effectively start, as will be explanied morein detail later.

The value of the function to be optimized, usually called the cost function, is evaluated at the position of the particle at each time step τ_k and it is modeled using the signal received by the receiver UAV at its current location $y_j : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}$, expressed in Eq.3.35.

The multiple-source seeking task is performed on a search-space Ω where the multiple victims are located, defined as the solution space where the y_j funtion is maximized.

$$\Omega = (x, y, z) \text{ s.t. } \begin{cases} X_{\min} \leq x \leq X_{\max}, \\ Y_{\min} \leq y \leq Y_{\max}, \\ Z_{\min} \leq z \leq Z_{\max} \end{cases}$$

where $X_{\min}$, $X_{\max}$, $Y_{\min}$, and $Y_{\min}$ denote the horizontal boundaries and $Z_{\min}$, $Z_{\max}$ denote the ground level and the highest limit respectively.

Assumption

We limit the search-space to the 2D plane as well, as we are not interested in determining the burial depth of the victims, which would need the emplyment of a different invariant from the NSS. Instead, we fix the z coordinate to the ground level for all the victims; while we assume a constant flying height of 3m for the UAVs, as already mentioned. Therefore, the search space becomes:

$$\Omega = (x, y, z) \text{ s.t. } \begin{cases} X_{\min} \leq x \leq X_{\max}, \\ Y_{\min} \leq y \leq Y_{\max}, \\ z = Z_{\text{fixed}} \end{cases}$$

The standard PSO algorithm iterative update of each particle's velocity (Eq. 3.36) and position (Eq. 3.37) in the search-space to improve its solution is defined as:

$$\begin{aligned} \mathbf{v}_j(\tau_k + 1) = \omega \mathbf{v}_j(\tau_k) + c_1 r_1 (\mathbf{pbest}_j(\tau_k) - \mathbf{p}_{r_j}(\tau_k)) + \\ + c_2 r_2 (\mathbf{gbest}(\tau_k) - \mathbf{p}_j(\tau_k)) \end{aligned} \tag{3.36}$$

$$\mathbf{p}_{r_j}(\tau_k + 1) = \mathbf{p}_{r_j}(\tau_k) + \mathbf{v}_j(\tau_k + 1) \tag{3.37}$$

where $\mathbf{pbest}_j$ represents the j-th particle's individual best position vector , $\mathbf{gbest}$ the global best position vector, ω is the inertia weight, c_1 and c_2 are positive constants, and $r_1, r_2 \sim U(0, 1)$ are random variables uniformly distributed over the interval $[0, 1]$.

A best previous position is where a particle obtains the maximum value of the output function in its search history. In 3.36, velocity $\mathbf{v}_j(\tau_k + 1)$ consists of three terms: the effect

of drone's previous velocity, effect of its best known position and effect of global best known position. While the inertia weight is critical in balancing global and local search. A larger inertia weight facilitates global exploration while a smaller one facilitates local exploitation.

In the standard PSO algorithm, there is only one global best particle, which means that only one optimal solution can be found. This limitation poses a challenge when dealing with our problem, where we are looking for multiple sources, and therefore, multiple local and global optima exist. To address this challenge, we employ the same strategy of the modified version of the PSO algorithm [28]. In this modification, the swarm is divided into M subpopulations, each tasked with exploring different regions of the search space. Following this idea, the *gbest* is referred to as the best particle of each subpopulation, not as the global best particle in the whole population. This means that these M best particles are separately able to catch different optima, at most M optima.

In addition, we perform further modifications to the PSO algorithm, we introduce physical constraints as already done in [31], an exclusion zone mechanism and a percistency of excitation Firstly, uniform random noise is added to the velocity of each particle to achieve a *persistence of excitation* state, allowing every drone to explore new and different directions, as described in Eq.3.38:

$$\mathbf{v}_j(\tau_k + 1) = \mathbf{v}_j(\tau_k) + \beta \cdot \mathbf{w} \cdot ||\mathbf{v}_j(\tau_k)|| \tag{3.38}$$

where:

- β is an hyperparameter chosen experimentally in the [0,1] range.
- $\mathbf{w} = [\mathrm{w}_1, \mathrm{w}_2]^\top$ and $\mathrm{w}_1, \mathrm{w}_2 \sim \mathcal{U}(-1, 1)$ represents a random vector where each component is uniformly distributed between -1 and 1.
- $||\mathbf{v}_j(\tau_k)||$ is the magnitude of the velocity vector of particle j.

Secondly, the UAVs are confined to the search-space and secondly their velocity is capped to a max value to avoid speed "explosion", [31]. The constraints are formulated as follows:

$$\begin{cases} \mathbf{p}_{r_j}(\tau_k + 1) = \mathbf{p}_{\text{lim}}, & \text{if } \mathbf{p}_{r_j}(\tau_k) \notin \Omega \quad (3.39) \\ \mathbf{v}_j(\tau_k + 1) = v_{\max} \dfrac{\mathbf{v}_j(\tau_k)}{\|\mathbf{v}_j(\tau_k)\|}, & \text{if } \|\mathbf{v}_j(\tau_k)\| > v_{\max} \quad (3.40) \end{cases}$$

where $\mathbf{p}_{\text{lim}}$ is the nearest point on the boundaries of the search space Ω, and $v_{\max}$ denotes the maximum allowed step length.
This means that:

- If the UAV exits the search space limits (i.e., $\mathbf{p}_{r_j}(\tau_k+1) \notin \Omega$), its position is corrected to the nearest boundary point of Ω in the next time step.
- If the magnitude of the velocity $\|\mathbf{v}_j(\tau_k + 1)\|$ exceeds the maximum allowable speed $v_{\max}$, then the velocity is scaled down to $v_{\max}$, while maintaining the same direction.

Exclusion Zone Mechanism

In order to avoid clustering around the same source and to ensure that the swarm does not converge only on a few sources, we introduce an exclusion zone mechanism. This strategy enforces additional positional constraints on the drones by leveraging the fact that each drone knows its current position in the search space.

A drone j defines an exclusion zone when it detects a source with a signal strength, the NSS, greater than a predefined threshold $\text{NSS}_{\text{thresh}}$. The exclusion zone is represented as a circular region, centered around the position of the drone, defined by:

$$\mathcal{E}_j = \{\mathbf{p} \in \mathbb{R}^3 \mid \|\mathbf{p} - \mathbf{c}_j\| \leq r_{\text{excl}}\},$$

where:

- $\mathbf{c}_j$ is the center of the exclusion zone, corresponding to the position of the drone when the source is detected ($\mathbf{p}_{r_j}(\tau_k)$).
- r_{excl} is the radius of the exclusion zone, chosen experimentally.

Instead, a drone j shares the center $\mathbf{c}_j$ and radius r_{excl} of its exclusion zone with neighboring drones if they are within the communication range r_{comm}. A neighboring UAV l receives the exclusion zone when:

$$\|\mathbf{p}_{r_l}(\tau_k) - \mathbf{p}_{r_j}(\tau_k)\| \leq r_{\text{comm}} \tag{3.41}$$

Each drone maintains a list of shared exclusion zones $\mathcal{E}_{\text{shared}}$, which is defined as:

$$\mathcal{E}_{\text{shared}} = \{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_j\}$$

which is an unordered and incomplete set of where the union is over all exclusion zones shared by neighboring drones.

Checking Each drone continuously checks whether its current position $\mathbf{p}_{r_j}(\tau_k)$ lies within any shared exclusion zone. This is verified by:

$$\mathbf{p}_{r_j}(\tau_k) \in \mathcal{E}_{\text{shared}}.$$

Avoidance If a drone j finds itself within a shared exclusion zone at time τ_k, it computes a new goal position $\mathbf{p}_{r_j}^{\text{goal}}(\tau_k + 1)$ to escape the zone. The avoidance mechanism follows these steps:

$$\mathbf{d}_j(\tau_k) = \mathbf{c}_j(\tau_k) - \mathbf{p}_{r_j}(\tau_k),$$

where $\mathbf{d}_j(\tau_k)$ is the direction vector pointing away from the exclusion zone center $\mathbf{c}_j(\tau_k)$, and $\mathbf{p}_{r_j}(\tau_k) \in \mathcal{E}_j(\tau_k)$.
The goal position the drone needs to reach in the next time steps is then calculated as:

$$\mathbf{p}_{r_j}^{\text{goal}} = \mathbf{p}_{r_j}(\tau_k) + \alpha \frac{\mathbf{d}_j(\tau_k)}{\|\mathbf{d}_j(\tau_k)\|} \tag{3.42}$$

where:

- α: Step size parameter that determines the distance the drone moves away from the exclusion zone.
- $\mathbf{p}_{r_j}^{\text{goal}}$: The computed goal position outside the exclusion zone.
- $\|\mathbf{d}_j(\tau_k)\|$: Norm (magnitude) of the direction vector $\mathbf{d}_j(\tau_k)$, which is used for normalization.

Once the drone reaches $\mathbf{p}_{r_j}^{\text{goal}}$, it resumes the PSO algorithm, continuing to search for new sources. This mechanism ensures the drone moves directly away from the exclusion zone while covering a distance proportional to α.

Chapter 4

Control

The purpose of this chapter is to model the dynamical system of each UAV and describe the control strategies used to achieve the desired position and velocity, while ensuring stability and robustness. The focus of our work is to localize the position of the avalanche victims, therefore the control task of the drones' flight is limited and interested mainly in controlling the altitude (mantaining a fixed height) and the horizontal positions x and y.

Each j-th UAV is modeled as a quadrotor vehicle, the most common multirotor aerial platform, characterized by four rotors attached to a rigid cross-shaped airframe. Each individual rotor generates both a forcet and a torque. The dynamics of each quadrotor are described using Euler-Newton laws of motion, with the motion expressed in terms of both the inertial frame F_g and the body frame F_b. As already mentioned, we have assumed the body frame F_b to coincide with the receiver frame F_r, relative to the the ARTVA equipment installed on the drone; therefore from now on we will denote the body frame with F_r.

The control of this underactuated system introduces more challenges compared to wheeled robots [19]. This section begins by presenting the fundamental notations and conventions, followed by the complete dynamical model of the UAV. A simplified model is then introduced, which assumes small angles to implement a Proportional-Integral-Derivative (PID) control strategy. Finally, a hierarchical control strategy used to regulate the quadrotor's position and velocity is discussed in detail, including both an outer position control in cascade with an inner attitude control mechanisms.

4.1 Notations

We introduce the notations and conventions used in this work to model the quadrotor dynamics and control. In particular, in Table 4.1 we introduce and summarize all the quantities and variables needed:

The laws of motion are described in both the global inertial frame F_g (particularly for translational dynamics) and the body frame F_r (in which the rotational dynamics are expressed more easily). The body right-hand frame F_r is fixed to the quadrotor (rotates with it), with its origin located at the center of mass. Its axes are defined as in [19]:

- The x-axis points forward along the quadrotor's direction of motion.

Table 4.1 UAV Quadrotor System Quantities

Symbol	Description	Unit
g	gravitational acceleration constant, $g \in \mathbb{R}_{\geq 0}$	m/s^2
m	mass of the quadrotor, $m \in \mathbb{R}_{\geq 0}$	kg
x_r	position of the quadrotor along x (east) in F_g	m
y_r	position of the quadrotor along y (north) in F_g	m
z_r	position of the quadrotor along z (up) in F_g	m
v_x	linear velocity along x in F_g	m/s
v_y	linear velocity along y in F_g	m/s
v_z	linear velocity along z in F_g	m/s
p	roll rate in F_b	rad/s
q	pitch rate in F_b	rad/s
r	yaw rate in F_b	rad/s
ϕ	roll angle (rotation about x-axis)	rad
θ	pitch angle (rotation about y-axis)	rad
ψ	yaw angle (rotation about z-axis)	rad
T	total thrust generated by the propellers, $T \in \mathbb{R}_{\geq 0}$	N

- The y-axis points to the left wing.
- The z-axis points upwards, consistent with the thrust direction.

For the inertial frame, we adopt a convention that does not use the standard NED[1] inertial reference frame, to ensure consistency throughout our work. Instead, the inertial frame F_g follows the ENU [2] convention (Figure 4.1), where the z-axis points upward, away from the center of the Earth. The inertial frame F_g is fixed to the ground, with its origin at a reference point (the takeoff location), and its axes are defined as:

- The x-axis points east.
- The y-axis points north.
- The z-axis points opposite the gravity vector.

The ENU local tangent plane is similar to NED, except for swapping "down" for "up" and x for y, which is more intuitive for certain geographical and aerial applications.
The orientation of the rigid body with respect to the inertial frame is described by the rotation matrix $\mathbf{R}_r^g(t) \in SO(3)$, which transforms vectors from the body frame F_r to the inertial frame F_g. The $\mathbf{R}_r^g(t)$ rotation matrix is obtained by successive rotations around the inertial frame axes, these rotations define the roll, pitch and yaw angles (ϕ, θ, ψ). We specify the order of rotations as $x - y - z$:

[1] NED stands for North-East-Down.
[2] ENU stands for East-North-Up.

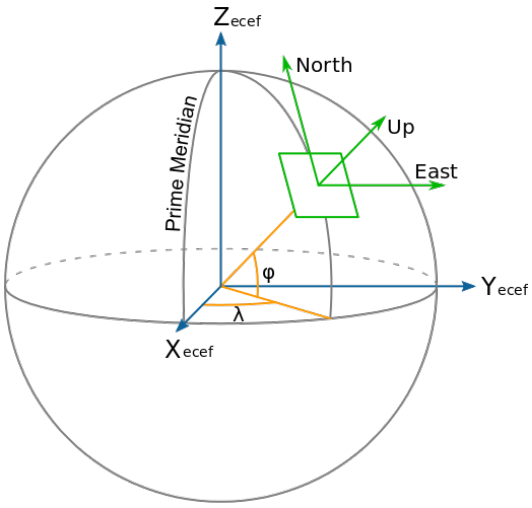

Figure 4.1 A diagram illustrating the ECEF (Earth-Centered, Earth-Fixed) and ENU reference frames. Adapted from a figure in [49].

- yaw (ψ): rotation about the z-axis of the inertial frame F_g.
- pitch (θ): rotation about the y-axis of the inertial frame F_g.
- roll (ϕ): rotation about the x-axis of the inertial frame F_g.

Since the successive rotations are relative to the fixed reference frame, the full rotation matrix $\mathbf{R}_r^g(t)$ is obtained as a product of the three individual rotation matrices:

$$\begin{aligned}
\mathbf{R}_r^g &= \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) \\
&= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \\
&= \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix}
\end{aligned} \tag{4.1}$$

Note
Considering $\mathbf{x} \in \mathbb{R}^3$, we define a square matrix to be skew-symmetric $\mathbf{S}(\mathbf{x}) \in \mathfrak{so}(3)$ if and only if:

$$\mathbf{S}(\mathbf{x})^\top + \mathbf{S}(\mathbf{x}) = 0 \tag{4.2}$$

Any rotation matrix $\mathbf{R}(t) \in SO(3)$ satisfies the orthogonality condition:

$$\mathbf{R}(t)\,\mathbf{R}^\top(t) = \mathbf{I} \tag{4.3}$$

Computing the time derivative on both sides of (4.3) gives:

$$\dot{\mathbf{R}}(t)\,\mathbf{R}^\top(t) + \mathbf{R}(t)\,\dot{\mathbf{R}}^\top(t) = 0 \tag{4.4}$$

This is the skew-symmetry condition (4.2) for a matrix defined as:

$$\mathbf{S}(t) \triangleq \dot{\mathbf{R}}(t)\,\mathbf{R}(t)^{\top} \tag{4.5}$$

By postmultiplying $\mathbf{R}(t)$ on both sides of 4.5 we obtain an expression for the variation in time of any rotation matrix:

$$\dot{\mathbf{R}}(t) = \mathbf{S}(t)\,\mathbf{R}(t) \tag{4.6}$$

Since $\mathbf{R}(t)$ is the product of basic rotations with respect to the fixed frame and denoting with $\boldsymbol{\omega}(t) = (\omega_x, \omega_y, \omega_z)^{\top} \in \mathbb{R}^3$ the angular velocity of the body frame with respect to the fixed frame, we can repeat the same process to find an expression of $\mathbf{S}$ which depends on the time derivative of the orientation angles, [13]:

$$\mathbf{S}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{4.7}$$

Furthermore, the angular velocity of the rotating body frame with respect to the fixed frame is related with the angular velocity expressed in the receiver frame as:

$$\boldsymbol{\omega} = \mathbf{R}_r^g\, {}^r\boldsymbol{\omega} \tag{4.8}$$

Then, we substitute Eq.4.8 in Eq.4.5 to obtain the the skew-symmetric matrix of ${}^r\boldsymbol{\omega}$, in the body frame:

$$\mathbf{S}(\boldsymbol{\omega}) = \mathbf{S}(\mathbf{R}_r^g\, {}^r\boldsymbol{\omega}) \tag{4.9}$$

Using the properties of skew-symmetric matrices:

$$\mathbf{S}(\mathbf{R}_r^g\, {}^r\boldsymbol{\omega}) = \mathbf{R}_r^g\, \mathbf{S}({}^r\boldsymbol{\omega})\, \mathbf{R}_r^{g\top} \tag{4.10}$$

Substituting this result (4.10) in Eq.4.6:

$$\dot{\mathbf{R}}_r^g = \mathbf{R}_r^g\, \mathbf{S}({}^r\boldsymbol{\omega})\, \mathbf{R}_r^{g\top}\, \mathbf{R}_r^g = \mathbf{R}_r^g\, \mathbf{S}({}^r\boldsymbol{\omega}) \tag{4.11}$$

We obtain the same expression of the inertial frame in the body frame.

4.2 Complete Quadrotor Dynamics

4.2.1 Translational Dynamics

Since the position of the UAV's center of gravity is represented by $\mathbf{p}_r \in \mathbb{R}^3$, expressed in the inertial frame F_g, we denote its time derivative as $\dot{\mathbf{p}}_r$ and the linear acceleration as $\ddot{\mathbf{p}}_r$.

Then, by applying Newton-Euler first law, we obtain the translational dynamics of the quadrotor in the fixed inertial frame F_g as in [4]:

$$m\ddot{\mathbf{p}}_r = \sum \mathbf{F}_{\text{ext}} = T\,\mathbf{R}_r^g\,\mathbf{e}_z - mg\,\mathbf{e}_z - \mathbf{d}_f \tag{4.12}$$

where $\mathbf{F}_{\text{ext}}$ denotes all the external forces acting on the vehicle, which includes the thrust force $T \in \mathbb{R}_{\geq 0}$ generated by the propellers and therefore represents the control force, the input of the system which can be controlled. It is aligned with the body z-axis by construction, as it is the sum of the single forces generated by the aligned rotors. It also includes the gravitational force mg, which instead is always directed towards the center of the Earth and therefore negative in our ENU fixed reference system. Finally, $\mathbf{d}_f$ represents the disturbance forces, accounting for exogenous effects such as aerodynamic drag and wind disturbances [35].

4.2.2 Rotational Dynamics

Newton-Euler's second law describes the variation in time of the angular momentum of a rigid body $\mathbf{L}$ in terms of the inertial reference frame, [11], [13]:

$$\frac{\mathrm{d}\mathbf{L}}{\mathrm{d}t} = \sum \boldsymbol{\tau}_{\text{ext}} \tag{4.13}$$

where $\mathbf{L} \in \mathbb{R}^3$ is the angular momentum of the rigid body, expressed in the fixed inertial frame F_g, and $\boldsymbol{\tau}_{\text{ext}} \in \mathbb{R}^3$ represents the total applied torque about the center of mass, which includes contributions from the control inputs, external disturbances, and aerodynamic effects.
The angular momentum $\mathbf{L}$ is related to the angular velocity as:

$$\mathbf{L} = \mathbf{J}\,\boldsymbol{\omega} \tag{4.14}$$

where $\mathbf{J} \in \mathbb{R}^{3\times3}$ is the inertia matrix of the rigid body, which is symmetric and positive definite ($\mathbf{J} = \mathbf{J}^\top > 0$).
However, it is more convenient to express the rotational dynamics in the body frame F_r of the rigid body, since the inertia matrix remains constant in this case. The opposite is not true, in fact the inertia matrix in the inertial frame is given by:

$$\mathbf{J} = \mathbf{R}_r^g\, {}^r\mathbf{J}\, \mathbf{R}_r^{g\top} \tag{4.15}$$

Therefore, we substitute this property of the inertia matrix (4.15) and Eq.4.8 in Eq.4.13, and use the result obtained earlier in 4.11:

$$\begin{aligned}\frac{\mathrm{d}(\mathbf{J}\boldsymbol{\omega})}{\mathrm{d}t} &= \dot{\mathbf{R}}_r^g\, {}^r\mathbf{J}\, \mathbf{R}_r^{g\top} {}^r\boldsymbol{\omega} + \mathbf{R}_r^g\, {}^r\mathbf{J}\, \dot{\mathbf{R}}_r^{g\top} {}^r\boldsymbol{\omega} + \mathbf{R}_r^g\, {}^r\mathbf{J}\, \mathbf{R}_r^{g\top} {}^r\dot{\boldsymbol{\omega}} \\ &= {}^r\mathbf{J}\, {}^r\dot{\boldsymbol{\omega}} + {}^r\boldsymbol{\omega} \times ({}^r\mathbf{J}\, {}^r\dot{\boldsymbol{\omega}})\end{aligned} \tag{4.16}$$

In the end, by comparing the right side of Eq. (4.16) with the right side of Eq.4.13, we obtain as in [35] the complete rotational motion of the rigid body:

$${}^r\mathbf{J}\, {}^r\dot{\boldsymbol{\omega}} + {}^r\boldsymbol{\omega} \times ({}^r\mathbf{J}\, {}^r\dot{\boldsymbol{\omega}}) = \sum \boldsymbol{\tau}_{\text{ext}} \tag{4.17}$$

The inertia matrix ${}^r\mathbf{J}$ depends on the geometry and mass distribution of the rigid body. For the quadrotor, assuming a rigid body structure with a dense spherical core of mass m_1 and radius r, along with four point masses m_2 located at a distance l from the center, the inertia matrix takes the diagonal form:

$${}^r\mathbf{J} = \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix},$$

where:

$$J_x = J_y = \frac{2}{5}m_1 r^2 + 2l^2 m_2, \quad J_z = \frac{2}{5}m_1 r^2 + 4l^2 m_2$$

Substituting Eq. (4.14) into Eq. (4.17), we obtain the rotational dynamics law:

$$ {}^{r}\mathbf{J}\,{}^{r}\dot{\boldsymbol{\omega}} + {}^{r}\boldsymbol{\omega} \times ({}^{r}\mathbf{J}\,{}^{r}\boldsymbol{\omega}) = \boldsymbol{\tau}_{\mathrm{u}} + \mathbf{d}_{\tau} \tag{4.18} $$

where $\boldsymbol{\tau}_{\mathrm{u}} \in \mathbb{R}^3$ is the controlled input torque vector dependent on the single torque generated by each rotor and with $\mathbf{d}_{\tau}$ we denote all the external disturbances.

In addition, we need to relate the angular velocity ${}^{r}\boldsymbol{\omega}$ in the body frame to the time derivatives of the roll, pitch and yaw angles (ϕ, θ, ψ) in the inertial frame. We assume that the variation of the angles $\dot{\phi}, \dot{\theta}, \dot{\psi}$ is small and therefore $\dot{\mathbf{R}}_x(\phi) = \dot{\mathbf{R}}_x(\theta) = \dot{\mathbf{R}}_x(\psi) = \mathbf{I}$:

$$ {}^{r}\boldsymbol{\omega} = \begin{pmatrix} p \\ q \\ r \end{pmatrix} = \begin{pmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{pmatrix} \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} \tag{4.19} $$

Finally, also the rotation matrix $\mathbf{R}_r^g(t)$ evolves in time as the rigid body rotates in space, following the result obtained earlier in Eq.4.11:

$$ \dot{\mathbf{R}}_r^g = \mathbf{R}_r^g\,\mathbf{S}({}^{r}\boldsymbol{\omega}) \tag{4.20} $$

where $\mathbf{S}({}^{r}\boldsymbol{\omega})$ is a skew-symmetric matrix in the form of (4.7), but relative to the body frame components:

$$ \mathbf{S}({}^{r}\omega) = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \tag{4.21} $$

4.2.3 Final Complete Quadrotor Dynamics

Combining the translational and rotational dynamics, the complete UAV dynamics is given by the system of equations 4.12, 4.20 and 4.18:

$$ \begin{cases} m\ddot{\mathbf{p}}_r = T\,\mathbf{R}_r^g\,\mathbf{e}_z - mg\,\mathbf{e}_z - \mathbf{d}_f \\ \dot{\mathbf{R}}_r^g = \mathbf{R}_r^g\,\mathbf{S}({}^{r}\boldsymbol{\omega}) \\ {}^{r}\mathbf{J}\,{}^{r}\dot{\boldsymbol{\omega}} = -{}^{r}\boldsymbol{\omega} \times ({}^{r}\mathbf{J}\,{}^{r}\boldsymbol{\omega}) + \boldsymbol{\tau}_{\mathrm{u}} + \mathbf{d}_{\tau} \end{cases} \tag{4.22} $$

More explicitly we can expand these equations and thanks to inverting Eq.4.19 and how we defined rotation matrix $\mathbf{R}_r^g$ in Eq.4.1, we obtain a system with twelve states and four control

inputs:

$$\begin{cases} \dot{x}_r = v_x \\ \dot{y}_r = v_y \\ \dot{z}_r = v_z \\ \dot{v_x} = -d_{f,x} + (\cos(\psi)\sin(\theta)\cos(\phi) + \sin(\psi)\sin(\phi))\frac{T}{m} \\ \dot{v_y} = -d_{f,y} + (\sin(\psi)\sin(\theta)\cos(\phi) - \cos(\psi)\sin(\phi))\frac{T}{m} \\ \dot{v_z} = -d_{f,z} - g + \cos(\theta)\cos(\phi)\frac{T}{m} \\ \dot{\phi} = p + \sin(\phi)\tan(\theta)q + \cos(\phi)\tan(\theta)r \\ \dot{\theta} = \cos(\phi)q - \sin(\phi)r \\ \dot{\psi} = \sin(\phi)\sec(\theta)q + \cos(\phi)\sec(\theta)r \\ \dot{p} = \tau_\phi + \frac{I_y - I_z}{I_x}qr + \frac{\tau_\phi}{I_x} \\ \dot{q} = \tau_\theta + \frac{I_z - I_x}{I_y}pr + \frac{\tau_\theta}{I_y} \\ \dot{r} = \tau_\psi + \frac{I_x - I_y}{I_z}pq + \frac{\tau_\psi}{I_z} \end{cases} \tag{4.23}$$

where $\tau_\phi, \tau_\theta, \tau_\psi$ are the components of input torque vector $\boldsymbol{\tau}_\mathrm{u} = (\tau_\phi, \tau_\theta, \tau_\psi)^\top$.
In a compact state-space form:

$$\dot{\boldsymbol{\xi}} = f(\boldsymbol{\xi}) + g(\boldsymbol{\xi})\boldsymbol{u} \tag{4.24}$$

where:

$$\boldsymbol{\xi} = [x_r, y_r, z_r, v_x, v_y, v_z, \phi, \theta, \psi, p, q, r]^\top$$
$$\boldsymbol{u} = [T, \tau_\phi, \tau_\theta, \tau_\psi]^\top$$

The non-linear dynamics of the state is desbribe by $f(\boldsymbol{\xi})$, while $g(\boldsymbol{\xi})$ represents the control input relationships with the system.

4.3 Simplified Dynamics

To facilitate control design, we consider a simplified model of the quadrotor dynamics, as proposed in [4]. This model assumes nominal conditions with no external disturbances and employs a small angle approximation for the roll (ϕ) and pitch (θ) angles. Additionally, we assume symmetric mass distribution and negligible aerodynamic and gyroscopic effects. Under these assumptions:

- Small angular velocities and small angles allow $(\dot{\phi}, \dot{\theta}, \dot{\psi}) \approx (p, q, r)$, simplifying the rotational dynamics.
- The Coriolis or gyroscopic term ${}^r\boldsymbol{\omega} \times (\mathbf{J}\,{}^r\boldsymbol{\omega})$ in the rotational dynamics is negligible for stabilization tasks, where angular velocities remain small.

The simplified dynamics are:

$$\begin{cases} m\ddot{\mathbf{p}}_r = T\,\mathbf{R}_r^g\,\mathbf{e}_z - mg\,\mathbf{e}_z \\ \mathbf{J}\,{}^r\dot{\boldsymbol{\omega}} = \boldsymbol{\tau}_{\mathrm{u}} \end{cases} \tag{4.25}$$

Explicitly expanded as:

$$\ddot{x}_r = (\cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi)\frac{T}{m} \tag{4.26a}$$

$$\ddot{y}_r = (\sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi)\frac{T}{m} \tag{4.26b}$$

$$\ddot{z}_r = \cos\theta\cos\phi\frac{T}{m} - g \tag{4.26c}$$

$$\dot{\phi} = \frac{\tau_\phi}{I_x} \tag{4.26d}$$

$$\dot{\theta} = \frac{\tau_\theta}{I_y} \tag{4.26e}$$

$$\dot{\psi} = \frac{\tau_\psi}{I_z} \tag{4.26f}$$

This simplified model reduces the complexity of the quadrotor dynamics, making it adapt to be controlled by simple control strategies such as PID[3] control.

4.4 PID Control

In this section, we implement a PID controller based on the simplified model from Section 4.26, but we aslo apply it to the full quadrotor dynamics described in Eqs.4.23, to evaluate its effectiveness in a real world scenario.

The primal objective is to regulate the altitude z and the horizontal positions x and y of the quadrotor along a specified trajectory $\mathbf{x}_d(t) \in \mathbb{R}^3$. In order to do so we define position error $\mathbf{e}(t) \in \mathbb{R}^3$ between the actual position $p_r(t)$ f the j-th UAV and the desired trajectory as:

$$\mathbf{e}(t) = \mathbf{x}_d(t) - \mathbf{p}_r(t) = \begin{bmatrix} e_x(t) \\ e_y(t) \\ e_z(t) \end{bmatrix} = \begin{bmatrix} x_d(t) - p_{r_x}(t) \\ y_d(t) - p_{r_y}(t) \\ z_d(t) - p_{r_z}(t) \end{bmatrix}$$

In Figure4.2, we define the block diagram which shows the hierarchical PID control of the full quadrotor dynamics.

4.4.1 Position Control

Focusing firstly on the altitude z, we drive the error esponentially to zero by controlling the acceleration vector of the quadrotors $\ddot{z}_r$ as in [4] to satisfy:

$$K_{p_z}(z_r - z_d) + K_{d_z}(\dot{z}_r - \dot{z}_d) + \ddot{z}_r - \ddot{z}_d = 0$$

[3] PID stands for Proportional-Integral-Derivative

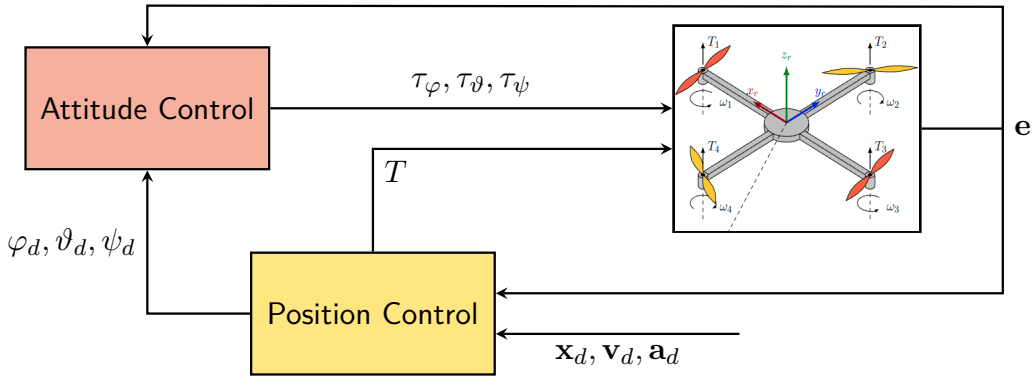

Figure 4.2 Block diagram of the PID control for trajectory tracking

Therefore, the virtual input of our control system is:

$$\ddot{z}_r = K_{p_z} e_z + K_{d_z} \dot{e}_z + \ddot{z}_d \tag{4.27}$$

where K_{p_z} and K_{d_z} are the proportional and derivative gains, respectively.
From the simplified dynamics (4.26c), the control thrust T required to maintain the altitude is given by:

$$T = \frac{m}{\cos\theta \cos\phi} (g + \ddot{z}_r) \tag{4.28}$$

Substituting $\ddot{z}_r$ from Eq.4.27 in Eq.4.28, we obtain:

$$T = \frac{m}{\cos\theta \cos\phi} (g + K_{p_z} e_z + K_{d_z} \dot{e}_z + \ddot{z}_d) \tag{4.29}$$

The control strategy we have adopted is hierarchical [19] since we have given priority to the altitude and horizontal positions control to determine the desired input thrust; only after we determine this quantity we define in cascade the control torques $\tau\theta, \tau\phi, \tau\psi$.

As before we want the position error to esponentially converge to zero, therefore we desire:

$$\begin{cases} K_{p_x}(x_r - x_d) + K_{d_x}(\dot{x}_r - \dot{x}_d) + \ddot{x}_r - \ddot{x}_d = 0 \\ K_{p_y}(y_r - y_d) + K_{d_y}(\dot{y}_r - \dot{y}_d) + \ddot{y}_r - \ddot{y}_d = 0 \end{cases} \tag{4.30}$$

where K_{p_x} and K_{p_y} are proportional gains, and K_{d_x} and K_{d_y} are derivative gains.
From the simplified translational dynamics, Equations 4.26a and 4.26b, we find the desired pitch θ_d and roll ϕ_d angles that ensure the system converges to the desired trajectory. We substitute the desired angles and obtain:

$$\frac{m}{T}\ddot{x}_r = \cos\psi \sin\theta \cos\phi + \sin\psi \sin\phi \tag{4.31a}$$

$$\frac{m}{T}\ddot{y}_r = \sin\psi \sin\theta \cos\phi - \cos\psi \sin\phi \tag{4.31b}$$

4.4.2 Attitude Control

After performing algebraic and trigonometric manipulations, we derive expressions for the desired roll and pitch angles:

$$\phi_d = \arcsin\left((T_y \cos\psi - T_x \sin\psi)\right) \tag{4.32}$$

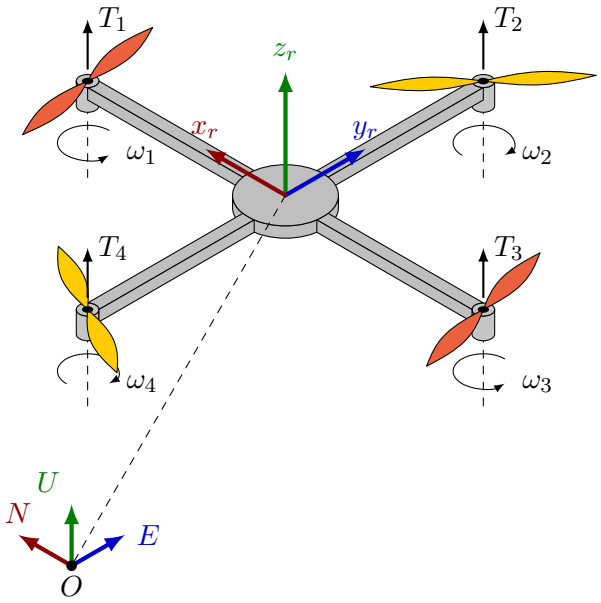

Figure 4.3 Quadrotor model

$$\theta_d = -\arcsin\left(\frac{T_x \cos\psi + T_y \sin\psi}{\cos\phi_d}\right) \tag{4.33}$$

where $T_x = \frac{m}{T}\ddot{x}_r$ and $T_y = \frac{m}{T}\ddot{y}_r$ are the desired components of the thrust vector in x and y and are found using Equations 4.31a and 4.31b respectively.

For the desired yaw angle, we use the relationship between the velocity components, so that the quadrotor mantains the correct heading direction. Specifically, the yaw angle is determined as:

$$\psi_d = \arctan 2(\dot{y}_d, \dot{x}_d), \tag{4.34}$$

Using the desired angles ϕ_d, θ_d, ψ_d, the control torques are defined as:

$$\tau_\phi = I_x\left(K_{p_\phi} e_\phi + K_{d_\phi}\dot{e}_\phi\right) \tag{4.35}$$

$$\tau_\theta = I_y\left(K_{p_\theta} e_\theta + K_{d_\theta}\dot{e}_\theta\right) \tag{4.36}$$

$$\tau_\psi = I_z\left(K_{p_\psi} e_\psi + K_{d_\psi}\dot{e}_\psi\right) \tag{4.37}$$

Where K_{p_ϕ}, K_{d_ϕ}, K_{p_θ}, K_{d_θ}, K_{p_ψ}, and K_{d_ψ} represent the proportional and derivative gains for roll, pitch, and yaw, respectively. While the input force was defined earlier in Eq.4.28.

4.4.3 Rotor Velocities

The configuration chosen for the drone to determine the relationship between the rotor speeds and the torques and thrust vector is the standard "+" configuration, as in [7], [11], [18], [19], [25]. In particular, we define and align the first rotor with the x_r axis of the body/receiver frame, while the rotor aligned with the y_r axis is defined as the second rotor, and we continue counting clockwise, as seen in Figure 5.6.

Each rotor generates a thrust T_i, where $i = 1, \ldots, 4$, which depends proportionally on the square of the rotor's angular velocity ω_i^2. Then, the total thrust is simply the sum of the thrusts generated by each individual rotor. On the other hand, the four spinning rotors create a yaw rotation (drag effect), which depends on whether their rotation is clockwise or counterclockwise. This is because the torque generated by each rotor acts in the opposite direction to the propeller's motion (angular speed). While the pitch and roll rotations depend on the arms length l and the two forces generated by the rotors on the two arms perpendicular to the respective rotation axis direction.

Each rotor's torque and thrust force depend on the drag coefficient $c_d \in \mathbb{R}_{\geq}$ and the thrust coefficient $c_t \in \mathbb{R}_{\geq}$ [18], which are parameters influenced by the geometry and size of the rotors and can be determined experimentally during static tests.

$$\begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} c_t & c_t & c_t & c_t \\ 0 & -c_t \cdot l & 0 & c_t \cdot l \\ c_t \cdot l & 0 & -c_t \cdot l & 0 \\ -c_d & c_d & -c_d & c_d \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \mathbf{A} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}$$

As already stated, the drone is an underactuated system, meaning that the 4 controllable inputs (the ω_i angular velocities) are fewer than the 6 DOF[4] the drone's movement has. In order to determine the rotor velocities ω_i^2, the matrix $\mathbf{A}$ must be invertible. Matrix $\mathbf{A}$ is invertible when $c_t, c_d, l \neq 0$, which is always true. Therefore, the matrix is always invertible as:

$$\begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \mathbf{A}^{-1} \begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \tag{4.38}$$

Then, the rotor velocities ω_i can then be obtained by taking the square root of the computed ω_i^2 values.

4.5 PID Control applied to PSO

Finally, in order to ensure that the drones in the PSO swarm follows the desired velocity and position commands we integrate the PID control described in the previous Section4.4 with the PSO algorithm described in Section3.2.4. The block diagram of the complete localization and control system is shown in Figure4.4.

Important
Note that the PSO algorithm updates are performed at a sample time $\tau_s \neq \tau_k$, where $s \in \mathbb{N}$, which is less frequent than the control time step τ_k. In particular, the PSO algorithm updates once every second, while $\tau_k \in \mathbb{R}$. This implies that at each sample time τ_s the NSS value is evaluated by each j-th receiver drone at $\mathbf{p}_{r_j}(\tau_s)$ using Eq.3.35. This value is used to update the personal best position $\mathbf{pbest}_j$ of each drone and the group best position $\mathbf{gbest}$ of each

[4]DOF stands for Degrees of Freedom

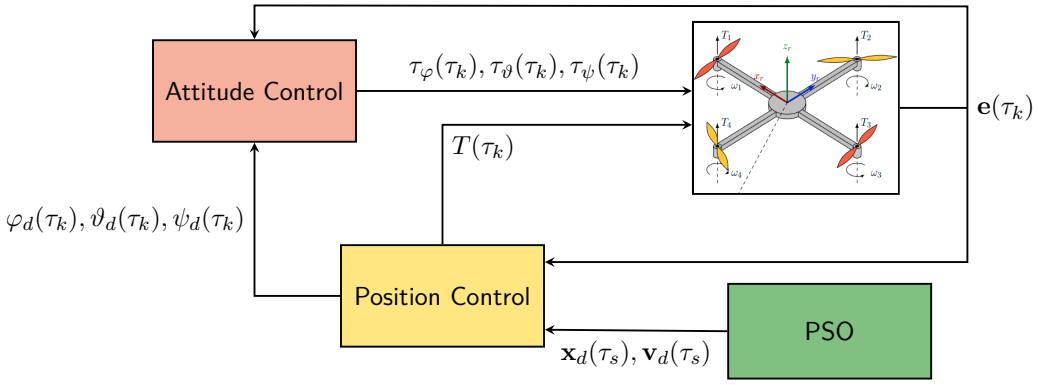

Figure 4.4 Block diagram of the PID control applied to the PSO algorithm

group, which occurs when the current NSS value exceeds previous records. Then, the drones update their 2D velocities and positions as in Eqs.3.36 and 3.37:

$$\begin{aligned}\mathbf{v}_j(\tau_s+1) = \omega\mathbf{v}_j(\tau_s) + c_1 r_1(\mathbf{pbest}_j(\tau_s) - \mathbf{p}_{r_j}(\tau_s)) + \\ + c_2 r_2(\mathbf{gbest}(\tau_s) - \mathbf{p}_j(\tau_s))\end{aligned} \tag{4.39}$$

$$\mathbf{p}_{r_j}(\tau_s+1) = \mathbf{p}_{r_j}(\tau_s) + \mathbf{v}_j(\tau_s+1) \tag{4.40}$$

In addition, the velocity clamping in Eq.3.40 is performed after this PSO update. We also note that the exclusion zone mechanism remains unvaried and follows the PSO algorithm in the same way.

The resulting $\mathbf{v}_j(\tau_s+1) = \mathbf{v}_d(\tau_s)$ and $\mathbf{p}_{r_j}(\tau_s+1) = \mathbf{x}_d(\tau_s)$ are then used as velocity and position desired commands for the PSO, when $\tau_s = \tau_k$. Then, during the next time interval $\Delta - s + 1s$ the PID control algorithm ensures the the the desired position and velocity are reached, and the drone's movement follows the full dynamics system defined in Eq.4.23:

$$\dot{\xi}(\tau_k) = f(\xi(\tau_k)) \tag{4.41}$$

where the drone's state ξ is updated at each time step τ_k in the interval $\Delta = \tau_{s+1} - \tau_s$. Additionally, the control dynamics ensure the search space boundary conditions as in Eq.3.39. This iterative alternation between PSO updates and independent drone control enables the swarm to balance global exploration and local exploitation, ensuring efficient source localization and actual control inputs.

Chapter 5

Experiments

In this chapter we present the implementation details of the experiments, the values of the employed parameters and the results obtained from the simulations. In particular, the parameters relative to the UAVs physical characteristics used in the simulations are from the real Dragonfly IV drone [6], these values are summarized in Table5.1.

Table 5.1 Drone Physical Parameters

Symbol	Value	Unit	Description
m	0.4	kg	Total mass of the drone
J_x	3.8278×10^{-3}	$\text{kg} \cdot \text{m}^2$	Moment of inertia about x-axis
J_y	3.8278×10^{-3}	$\text{kg} \cdot \text{m}^2$	Moment of inertia about y-axis
J_z	7.1345×10^{-3}	$\text{kg} \cdot \text{m}^2$	Moment of inertia about z-axis
l	0.205	m	Distance from center to rotor
c_t	6.354×10^{-4}	–	Coefficient for thrust generation
c_d	0.048	–	Coefficient for torque generation

5.1 Implementation

The implementation of this work is done in *Matlab* and it is structured into multiple sections and files, to increase the modularity and readability of the code while simplifying the debugging process at the same time. The entire code is divided into the following categories:

- A folder containing the **RLS** algorithm in the case where the sources do not influence one another;
- A folder containing the complete **PSO** algorithm integrated with the PID control (the main focus of our work);
- A file with the complete **NSS** analysis of rotational invariance and symmetry properties;
- A file with the **PID** control applied to tracking the initial trajectory the drones follow for setup.

Th latter contains the same controller and model functions of the `Controller` class of the PSO algorithm. The only difference consists in the desired velocities and positions, which are given by a predefined trajectory and by the PSO algorithm respectively.

In both the last two experiments (trajectory tracking and PSO with control) the controller gains are chosen with values as described in Table 5.2:

Table 5.2 Control Gains for Position and Attitude Control

Proportional Gain (k_p)		**Derivative Gain (k_d)**		**Control Type**
Symbol	**Value**	**Symbol**	**Value**	
k_{xp}	0.1	k_{xd}	0.5	Position x
k_{yp}	0.1	k_{yd}	0.54	Position y
k_{zp}	1.0	k_{zd}	2.5	Position z
$k_{p\phi}$	2.0	$k_{d\phi}$	2.5	Roll ϕ
$k_{p\theta}$	2.0	$k_{d\theta}$	2.5	Pitch θ
$k_{p\psi}$	2.0	$k_{d\psi}$	4.0	Yaw ψ

The complete PSO with Control algorithm has the following main components:

- Setup and `main loop` of the simulation;
- The `Particle` class;
- The `Controller` class;
- The `ARTVAs` class;
- The `Plotter` class.

The main structure of the simulation is the following: we first perform a setup step in which we initialize the drones, the transmitting ARTVAs and other useful variables and constants. Then, the simulation is divided into two phases:

- **Exploration** phase: the drones are asked to position themselves uniformly in the search space by followying a predifined trajectory;
- **Exploitation** phase: the drones are moved by the modified complete PSO algorithm to localive the multiple avalanche victims (ARTVAs).

If we have successfully estimated the position of all the trasmitting ARTVAs we stop the simulation, otherwise if any of the victims cannot be found the simulation stops after a predefined (`n_iterations`) maximum number of iterations.

The implementation begings with the set-up of many variables and constants that will be useful throughout the whole experiment (see Table 5.3); and the initialization of the classes and theirs crucial parameters (UAV physical quantities, controller gains and particles' data).

Table 5.3 PSO Hyperparameters

Symbol	Value	Unit	Description
$n_{\text{iterations}}$ (`n_iterations`)	450	–	Number of iterations
ω (`inertia`)	1.05	–	Inertia weight
c_1 (`cognitive_factor`)	2.05	–	Cognitive factor
c_2 (`social_factor`)	1.5	–	Social factor
`NSS_threshold`	10^9	–	NSS signal locating threshold
β (`velocity_randomness`)	0.6	–	Velocity randomness
$\Omega_{x,y}$ (`bounds`)	$[-80, 80]$	m	Search space limits
v_{max} (`max_velocity`)	2	m/s	Maximum velocity
r_{comm} (`communication_radius`)	10	m	Communication radius
α (`step_size`)	90	m	Move away distance
r_{excl} (`exclusion_zone_radius`)	3	m	Exclusion zone radius
δt (`dt`)	0.04	s	Control time step
Δ (`iteration_duration`)	1	s	PSO iteration duration

The most important hyperparameters include the number of drones (`n_drones`), the number of iterations (`n_iterations`), search space boundaries (`bounds`), and maximum velocity (`max_velocity` or v_{max}). The sources (`p_sources`) are fixed at predefined positions, and their number (`n_sources`) determines how the drones are grouped during initialization, therefore we suppose we have at least a prior knowledge of the maximum number of victims. Instead, the PSO algorithm employs the inertia (`inertia` or ω), cognitive (`cognitive_factor` or c_1), and social (`social_factor` or c_2) factors to balance exploration and exploitation, ensuring drones explore the search space efficiently before converging to optimal solutions. Other relevant parameters include the `velocity_randomness` (β), which determines the level of percistency of excitation, the number of groups (`n_groups`) which matches the number of sources and the `communication_radius` (r_{comm}), which sets the maximum range for drone data transfer. Instead, the `step_size` (α) defines the meters the drones need to travel to move away from an exclusion zone they entered. The `exclusion_zone_radius` (r_{excl}) prevents redundant exploration near detected sources and avoids multiple drones converging on the same source. It is important in determining the "resolution" of the algorithm, since no more than one victim can be found in that radius by the drones, only by the human rescuers. Instead, the `NSS_threshold` has been carefully finetuned so that the drones are precise enough in determing the value of the victims, but also fast enough in creating a n exclusion zone so that no other drones "get stuck" around the same source. Time step (`dt`) represents the τ_s control time, while the `iteration_duration` is the Δ of 1 second in Section4.5, determining the interval between one PSO update and another. Each drone is has an initial position, velocity, and group affiliation, while a data structure is preallocated to store its trajectory throughout the search process. In addition to this, we also initialize the `Plotter` class, whose job is to display in real time drone positions, trajectories, and exclusion zones in real time.

Then the two phases begin: the exploration phase uses a radial pattern to evenly partition the search space, dividing the space unfiformly between the drones. They move in straight lines toward their goals at maximum allowed speed.

Instead, during the main PSO loop, drones evaluate their positions based on the NSS signal received from the `ARTVAs` class. If a drone detects NSS values higher than a (`NSS_threshold`), it flags the localization of a drone. Drones that enter exclusion zones move away from it and create a new group, with their velocities, personal best (`p_best` or $\mathbf{pbest}$), and group best (`g_best` or $\mathbf{gbest}$) resetted.

After the real time simulation stops visualizing the final positions of drones and sources, plotting their trajectories, and computing errors between detected and true source positions.

5.1.1 `ARTVAs` class

DA RIGUARDARE SE SI DECIDE DI CAMBIARE (M. OPPURE NO?) The sole purpose of the `ARTVAs` class is to emit the signal received by a drone, the NSS. The signal is "emitted" using the *send_signal_NSS()* function, which takes the position of the drone as input and returns a single floating-point value representing the perceived signal strength, the same in Eq.4.23.

The main parameter of this class is `C`, a scaling factor that determines the magnitude of the NSS signals, whoch is set to a very high value of 10^8, since we do not consider the other physical parameters involved and the signal has very low values with increasing distances.

DA RIGUARDARE SE SI DECIDE DI CAMBIARE (M. OPPURE NO?) It calculates both the combined signal strength from all sources and the individual contributions based on how far a drone is from each source. These calculations ensure that drones receive realistic signal strengths during the optimization process. The *superpositionNSS()* method computes the total NSS received by a drone by summing the signals from all sources, considering their distances from the drone. The *send_signal_NSS()* method calculates the NSS from a specific source to a given drone position. This is done using an inverse fourth-power relationship with distance, ensuring the signal weakens realistically as the drone moves farther from the source. The *send_signal_NSS()* method applies the formula described in Equation **??**, where r is capped at a minimum value of 10^{-5} to prevent division by zero. The NSS value is then calculated according to the formula in Equation **??**.

5.1.2 `Plotter` class

The `Plotter` class is the component that is responsible for the visualization of the simulation in real time during: it opens a separate window at the beginning of the simulation and efficiently displays the simulation inside it. It also provides methods for initializing plots, updating drone positions, displaying exclusion zones, and presenting trajectories. This visualization aids in monitoring the algorithm's progress and understanding group dynamics. This class naturally includes properties to manage the figure sizes, axes limits, drone and

source markers, color schemes, legends, and bounds of the plot. It also updates the plot title with the current iteration number and simulation time.

The *draw()* method dynamically updates the positions of the drones and displays the fixed sources positions. Since the draw calls of the `Plotter` can be the real bottleneck in the main loop of the simulation, we carefully optimized the code relative to this task so that redrawing the scene only updates the parts that changed between subsequent time-steps and everything else is kept as is.
Note that the simulation waits a defined time-step at each loop, from which all the computational time is subtracted, so to have faster simulations. However, this does not affect the time used to comment the results or for other calculations used in the algorithm, which remains the actual time it takes the drones to localize the victims. After this optimization, we managed to make the `Plotter` run fast enough that the simulation seems effectively in real time (since the draw calls do not run in a separate thread and therefore are blocking calls).

In particular, the `Plotter` displays:

- The exploration phase trajectory lines in radial shape with respect to a imaginary circular boundary;
- The drones, each assigned their own unique color based on their group affiliation;
- The positions of the real ARTVAs;
- The position of the multiple estimations of the ARTVA, each with the same color of the corresponding group;
- The exclusion zones, represented as circles whose centers are determined by the drone's position when the signals surpassed the threshold.

Note that when a new group is created during the simulation, a new color is dynamically generated and added to the palette. The drone colors and the plot legend are updated accordingly.

In the final stages of the simulation, the *plot_best()* method displays the best estimates for source positions achieved by each group. These estimates are marked with group-specific colors, and the legend is updated to include them. Similarly, the *plot_trajectories()* method visualizes the paths traveled by all drones, providing insight into their trajectories during the PSO algorithm, since during the real-time simulation the positions are shown only at each PSO time step τ_s.

5.1.3 `Controller` class

The `Controller` class contains the complete quadrotor dynamics and the PID control laws. We define in it the physical properties of the drone, such as the mass, the inertia matrix, and the torque and drag coefficients (see Table 5.1). Furthermore, we define the control proportional and derivative gains both for the x, y, z positions and the the roll (ϕ) and pitch (θ) angles.

The *quadrotor_full_dynamics()* method computes the drone's translational and rotational dynamics based on its current state, external forces and torques. The state derivative is computed using the Equations4.23.

A second method, *control_laws()*, is responsible for calculating the input force and torques required to control the quadrotor. This method first computes the position and velocity errors, which are used to determine the desired thrust like in Eq. 4.29. From this desired thrust, the method calculates its x and y components T_x and T_y, which are used to derive the desired roll ϕ_d and pitch θ_d angles with Eqs. 4.32, 4.33 respectively. The desired yaw torque is computed separately to align the quadrotor with the desired heading, ensuring accurate orientation control. These desired angles and the associated angular velocity errors are then combined with the controller gains to calculate the commanding input torques, following Eqs. 4.35, 4.36, 4.37.

Finally, the *compute_rotor_velocities()* method determines the rotor speeds needed to achieve the commanding thrust and torques. The speeds are found inverting $\mathbf{A}$ matrix as in Eq.4.38. The computed rotor speeds are checked to ensure non-negative values, as negative speeds are physically invalid.

5.1.4 `Particle` class

The `Particle` class, which is also called `Drone` class, models each drone in the conplete PSO algorithm integrated with the PID control and postioning phase.
Each particle stores its current position and velocity, as well as its personal best position (`p_best`), the corresponding highest NSS value found so far (`nss_best`), their group index (`group_idx`), the `victim_found_flag` to remember the found source and data structures to remember the exclusion zones that have been shared with and by the drone. The constructor of the class initializes the position at the center of the search space, and the personal best to this position.

This class contains the most important methods: the *update_velocity()* method modifies the particle's velocity using the formula described in 4.39. Then, random noise is added to the velocity as in Eq.3.38, while constraints ensure the velocity does not exceed the `max_velocity` following Eq. 3.40.

Instead, the *update_state()* function is the one responsible of the implementation of the complete quadrotor dynamics and control laws, by calling the respective methods we have seen in the `Controller` class5.1.3. This is achieved through the initialization of a `Controller` class in every `Drone` class.

Furthermore, the *evaluate_nss()* method calculates the NSS value at the particle's current position by passing the `ARTVAs` class approriate function, as described in Algorithm1. Its personal best is updated if higher then the previous best. and if the particle discovers a source it sets the flag to create an exclusion zone to true. The exclusion zone is mantained until the end of the simulation. In this last case the inertia is reduced to decrease the excitation level.

Algorithm 1: `evaluate_nss` (MATLAB function)

Input: $\mathbf{p}_r$: current position of the particle.
Input: `p_sources`: array of source positions.
Input: $superpositionNSS$: function to compute the total NSS at a given position.
Output: `nss_value`: NSS value at the particle's current position.
Output: `p_best`: personal best position.
Output: `nss_best`: personal best NSS value.
Output: `my_exclusion_zone`: center of the created exclusion zone.

Function `evaluate_nss(`$\mathbf{p}_r$, *`p_sources`*`): /*` `*/`

Compute the NSS value at the current position:

1: `nss_value` $\leftarrow$ *superpositionNSS*($\mathbf{p}_r$, `p_sources`);
2: **if** `nss_value` $>$ `nss_best` **then**
3: `p_best` $\leftarrow \mathbf{p}_r$;
4: `nss_best` $\leftarrow$ `nss_value`;
5: **end**
6: **else if** `nss_value` $>$ `NSS_threshold` **and** `victim_found_flag` = `false` **then**
7: `victim_found_flag` $\leftarrow$ `true`;
8: `my_exclusion_zone` $\leftarrow \mathbf{p}_r$;
9: `inertia` $\leftarrow 0.5$;
10: **end**
11: **end**

Communication between particles is allowed only by the *can_I_communicate_with()* method, which determines if two particles are within the predifined communication radius, which is condition 3.41.

The *share_exclusion_zones()* method allows the drone to communicate to its neighbouring drones the position of the found "supposed" source.

Finally, the *check_if_in_exclusion_zone()* method verifies if a particle is inside any exclusion zone the drones knows of and the *move_away_from_exclusion()* function makes the drone take big "jump" in the same incoming direction to get as far as possible from the zone, the goal position is computed using Eq.3.42.

5.1.5 `Main Loop`

The first step in the `main loop` involves the initialization of the UAVs (particles) and in particular the sorting of the drones in different groups as described in Algorithm2. The drones are grouped firstly by filling all the available sposts, then by adding randomly the drones in excess to the gruops and then they are sorted for improved performances.
Note that the function `sort_drones_in_groups` ensures the creation of sufficient groups for the localization of multiple sources only when the number of drones is equal to or greater than the number of sources.

Algorithm 2: `sort_drones_in_groups` (MATLAB function)

Input: `n_drones`: number of drones available for assignment.
Input: `n_sources`: number of sources to which drones need to be assigned.
Output: `group_indices`: array of indices indicating the assigned group.
Output: `n_groups`: the obtained number of groups.

Function `sort_drones_in_groups(n_drones, n_sources): /*` `*/`

1: **if** `n_drones` $>$ `n_sources` **then**
2: **Assign** at least one drone to each group, following mathematical order;
3: **Assign randmly** remaining drones to existing groups;
4: **end**
5: **else if** `n_drones` = `n_sources` **then**
6: **Assign** each drone to a unique group;
7: **Sort** the group indices to ensure orderly grouping;
8: **end**
9: **else**
10: **Assign orderly** drones up to `n_drones`;
11: **end**
12: **Set** the number of groups: `n_groups` $\leftarrow$ `max(group_indices);`
13: **end**

Exploration phase

At the start of the simulation, all drones are located at the origin of the search space. During the first Exploration Phase, the PSO algorithm is not yet employed. During this phase, the drones move according to predefined trajectories at maximum speed $v_{\max}$. The trajectories are determined based on the number of drones and follow a radial pattern.The drones follow these trajectories until they have covered a distance `travel_distance` equivalent to half the boundary of the search space. The goal each drone needs to reach is computed using the following Algorithm 3:
where `angle_step` is the angular increment to uniformly divide the space, `drone_angle` represents the angle assigned to each drone, while `m` is the slope of the trajectory calculated from the drone's angle.

Exploitation Phase

This is the most important part of the simulation since it involves the localization of the victims through the deployment of our modified PSO algorithm with the PID control, described in Section 4.5.

At each τ_s iteration step until `n_iterations`, each j-th UAV checks the neighbouring drones with the *can_I_communicate_with()* function and with a positive response they share their exclusion zones with each other (if they have found a source). The communication radio (r_{comm}) is set to 10 m.

Then, it calls *check_if_in_exclusion_zone()* to verify if it's position is inside the radius of any of the saved exclusion zones. In case of a positive response, the drone is commanded to

Algorithm 3: `exploration_goal` (MATLAB function)

Input: `n_drones`: number of drones available for exploration.
Input: `bounds`: maximum limit of the search space.
Output: `goals`: array containing x,y positions of the exploration goals.

Function `exploration_goal(n_drones, bounds): /*` `*/`

1: **Initialize** an empty array `goals`.
2: **Set** `angle_step` $\leftarrow \frac{360}{\texttt{n\_drones}}$.
3: **Set** `travel_distance` $\leftarrow \frac{\texttt{bounds}}{2}$.
4: **for** drone $j = 1$ **to** m **do**
5: **Compute** angle for the current drone: `drone_angle` $\leftarrow j \cdot$ `angle_step`;
6: **Calculate** `slope` $\leftarrow \tan(\textit{deg2rad}(\texttt{drone\_angle}))$;
`/* 1st quadrant */`
7: **if** `drone_angle` > 315 **or** `drone_angle` ≤ 45 **then**
8: **Set** `goal` $\leftarrow [\texttt{travel\_distance}, \texttt{slope} \cdot \texttt{travel\_distance}]$
`/* 2nd quadrant */`
9: **else if** `drone_angle` > 45 **and** `drone_angle` ≤ 135 **then**
10: **Set** `goal` $\leftarrow [\frac{\texttt{travel\_distance}}{\texttt{slope}}, \texttt{travel\_distance}]$
11: **end**
`/* 3rd quadrant */`
12: **else if** `drone_angle` > 135 **and** `drone_angle` ≤ 225 **then**
13: **Set** `goal` $\leftarrow [-\texttt{travel\_distance}, \texttt{slope} \cdot -\texttt{travel\_distance}]$
14: **end**
`/* 4th quadrant */`
15: **else if** `drone_angle` > 225 **and** `drone_angle` ≤ 315 **then**
16: **Set** `goal` $\leftarrow [-\frac{\texttt{travel\_distance}}{\texttt{slope}}, -\texttt{travel\_distance}]$
17: **end**
18: `goals` $\leftarrow$ `goals` + [`goal`];
19: **end**
20: **end**

execute the *move_away_from_exclusion()* function. Then, its velocity is reset to zero, and its personal best position, `p_best`, is assigned to a random point far from the new location. Additionally, the $\mathbf{gbest}$ is reset to $-\infty$, and the drone's `nss_best` is also reset to $-\infty$, ensuring the drone restart exploring with a fresh start. For simplicity reasons, this movement is considered "instantaneous", meaning that in the simulation, the drone's journey to the goal is not displayed. Instead, in the next iteration, it will already be at the designated location. However, the number of iterations and the simulation time account for the actual time required for the drone to reach the goal at maximum speed. In case of a negative response, the *evaluate_nss()* is called and the group best ($\mathbf{gbest}$) is updated if the received NSS signal is greater than the previous best.

Subsequently, the *update_velocity()* method computes the desired velocity command for the control loop and the desired position is obtained using the standard PSO formula in Eq. 4.40. At each time step τ_k until the end of the Δ, the *update_state()* function is called and

the boundries are enforced as in 3.39. Finally, we can present the complete modified PSO algorithm in Algorithm 4. Note that the inertia of a drone that has found a source is set to zero ($\omega = 0.5$), increasing convergence speed. The final output is the set of best estimates `g_best_values`, composed of the best estimated position of a source by each group.

Algorithm 4: Particle Swarm Optimization for Multi-Source Localization

Initialize positions of drones to the center of the search space.
Initialize `g_best_values`.
Assign each drone to a group using the function in Algorithm 2.

Exploration Phase:
Compute exploration goals using the function in Algorithm 3;
for drone $j = 1$ **to** m **do**
    Move to goal with `max_velocity`;
end

Exploitation Phase:
for $\tau_s = 1$ to `n_iterations` **do**
    for drone $j = 1$ **to** m **do**
        for each other drone $l \neq j$ **do**
            if *can_I_communicate_with()* **or** `victim_found_flag` = true **then**
                Share exclusion zones between j and l;
            end
        end
        if *check_if_in_exclusion_zone()* = true **then**
            if drone j is not alone in its group **then**
                Reassign drone j to a new group;
                Initialize new `g_best` and `nss_best`;
            end
            Move away from exclusion zones;
            Reset personal best `p_best`;
            Set drone j `inertia` to 0.5;
        end
        else
            Evaluate NSS at $\mathbf{p}_j(\tau_s)$;
            if `nss_value` $>$ `nss_best` **then**
                Update `p_best`;
            end
            if `nss_value` $>$ `g_best` **then**
                Update `g_best_values`;
            end
        end
        Compute desired velocity: *update_velocity()*;
        Apply persistence of excitation as in Eq.(3.38);
        Limit velocity to `max_velocity` as in Eq.(3.40);
        Desired position: as in Eq.(4.40);
        Set `iteration_duration` to 1;
        for `t` $= 0$: `dt` : `iteration_duration` **do**
            Control dynamics: *update_state()*;
            Enforce boundaries on position as in Eq.(3.39);
        end
    end
end

5.2 Results

5.2.1 PID for Trajectory Tracking

The results of PID control using the full quadrotor dynamics model (4.22), applied to trajectory tracking during the first positioning (exploration) phase of the algorithm, are presented in the following figures: Figure 5.1 illustrates the comparison between the actual and desired trajectory in 3D space, showcasing the effectiveness of the control strategy in aligning the drone's path with the desired trajectory.

The desired trajectory chosen for this phase is designed to a provide smooth and easily controlled movement. The x and y components of the trajectory follow a linear motion at constant maximum velocity v_{max}. Since the drones need to travel on straight lines that radially distribute over the horizontal xy-plane, we define the `drone_angle` in Alg. 3 as γ the orientation w.r.t. the x-axis of the trajectory. The altitude is modeled to rise smoothly using an exponential function, achieving a desired altitude z_d, with a smoothness factor k controlling the rate of ascent. The mathematical expressions for the desired trajectory and its derivatives are:

$$\begin{aligned}
\mathbf{x_d}(t) &= \begin{bmatrix} x_d(t) \\ y_d(t) \\ z_d(t) \end{bmatrix} = \begin{bmatrix} v_{\text{max}}\, t \cos(\gamma) \\ v_{\text{max}}\, t \sin(\gamma) \\ z_d \left(1 - e^{-kt}\right) \end{bmatrix} \\
\dot{\mathbf{x}}_\mathbf{d}(t) &= \begin{bmatrix} \dot{x}_d(t) \\ \dot{y}_d(t) \\ \dot{z}_d(t) \end{bmatrix} = \begin{bmatrix} v_{\text{max}} \cos(\gamma) \\ v_{\text{max}} \sin(\gamma) \\ z_d k e^{-kt} \end{bmatrix} \\
\ddot{\mathbf{x}}_\mathbf{d}(t) &= \begin{bmatrix} \ddot{x}_d(t) \\ \ddot{y}_d(t) \\ \ddot{z}_d(t) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -z_d k^2 e^{-kt} \end{bmatrix}
\end{aligned} \tag{5.1}$$

Figures 5.2, 5.3, and 5.4 provide a detailed breakdown of the trajectory tracking along the x, y, and z axes, respectively. These plots highlight the individual tracking performance in each dimension and confirm the controller's precision in following the desired positions.

Figure 5.5 shows the simulation over time, providing an overview of the control system's behavior during this phase. Additionally, the rotor velocities calculated using (4.38) are plotted in Figure 5.6.

The results demonstrate that after an initial overshooting the position controller is able to follow the desired trajectory. In particular, altitude control (hovering) is also quite important in our appplication and it is successful in our results. Furthermore, the rotor velocities exhibit regular and achievable values, meaning the actuation inputs can be followed. These results were obtained under a nominal scenario, where wind disturbances or other environmental factors are not considered, since the primary focus of this work is not on the control robustness.

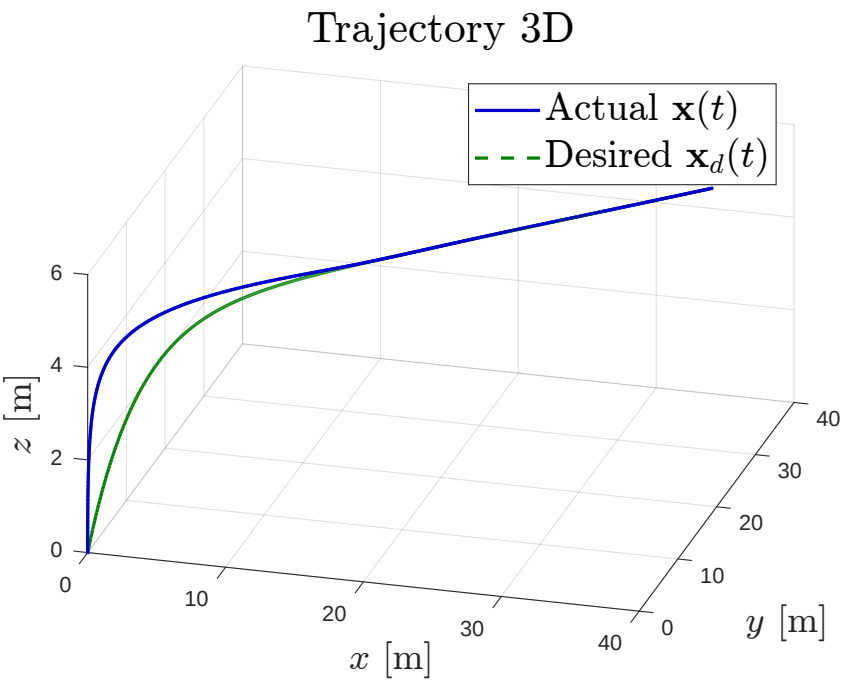

Figure 5.1 Comparison of the actual and desired trajectory in 3D space during the exploration phase.

5.2.2 PSO wih Control

In the context of search and rescue (S&R) missions, the number of available UAVs is typically limited (ranging from 1 to 10 [3]), and the search area usually spans only a few thousand square meters. In order to have faster simulations and more constrained values we limit the search space to the hundreds of meters. However, the algorithm is scalable to bigger distances while mantaining the same performances by incrementing the drones' speed, or enhancing the capabilities of the ARTVA receiver, which is usually limited to our chosen limits.

Each drone in the PSO control integrated framework is initialized with the hyperparameters described in the previous sections. The drones start their search with an initial position at the center of the search space, defined as $[0, 0]$. Their initial velocities are randomized using the velocity randomness factor capped at a maximum velocity of 2 m/s to prevent overshooting targets. The search space is constrained within bounds of $[-80, 80]$ m for both the x and y directions.

Each drone's best-known position ($\mathbf{pbest}$) is initialized to its starting position, while its best NSS value is set to $-\infty$, ensuring that any detected signal improves its current state. The drones operate with an inertia weight of 1.05, balancing exploration and exploitation during the optimization process.

The drones are grouped based on a variable group index, and each group best-known position ($\mathbf{gbest}$) is also initialized to $-\infty$. reagarding the orientation and UAVs dynamics, the roll, pitch, and yaw angles are initialized to $[0, 0, 0]$. The initial velocity during the second phase of the algorithm is the same of the arriving velocity at the goal of the first phase. These parameters collectively define the initial state and behavior of the drones in

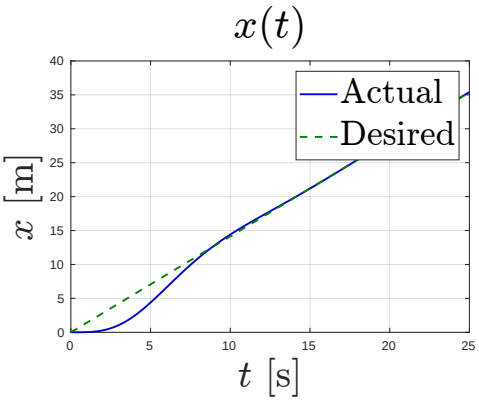

Figure 5.2 Actual and desired $x(t)$.

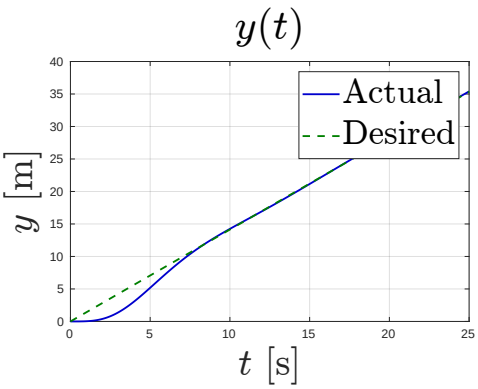

Figure 5.3 Actual and desired $y(t)$.

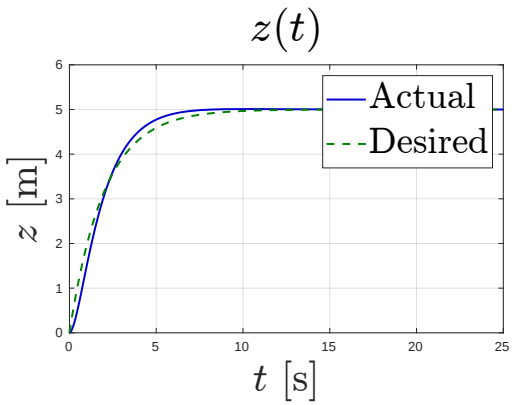

Figure 5.4 Actual and desired $z(t)$.

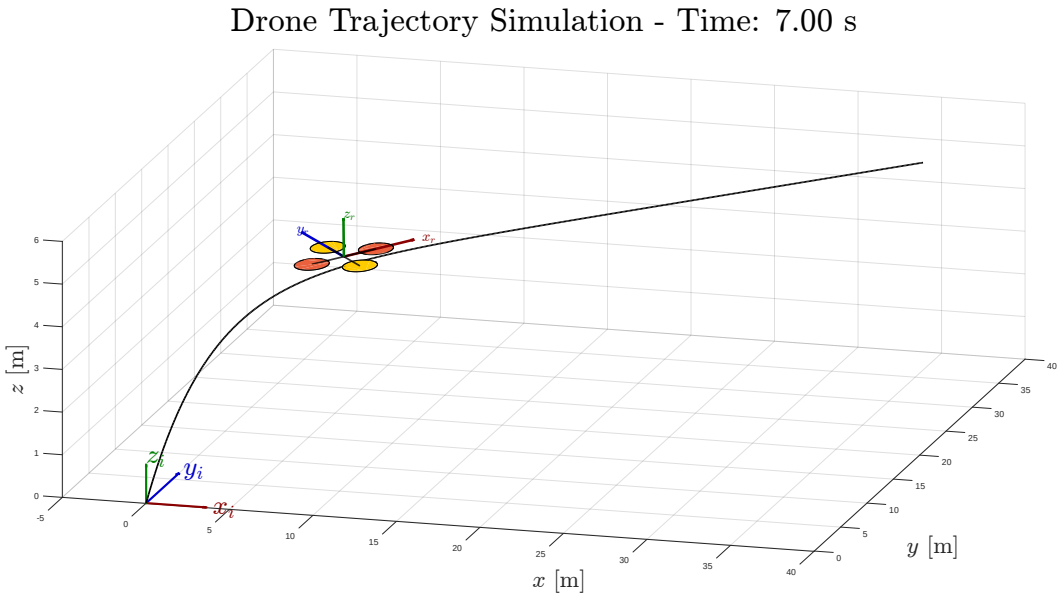

Figure 5.5 Time instant during the simulation illustrating the system's dynamic behavior.

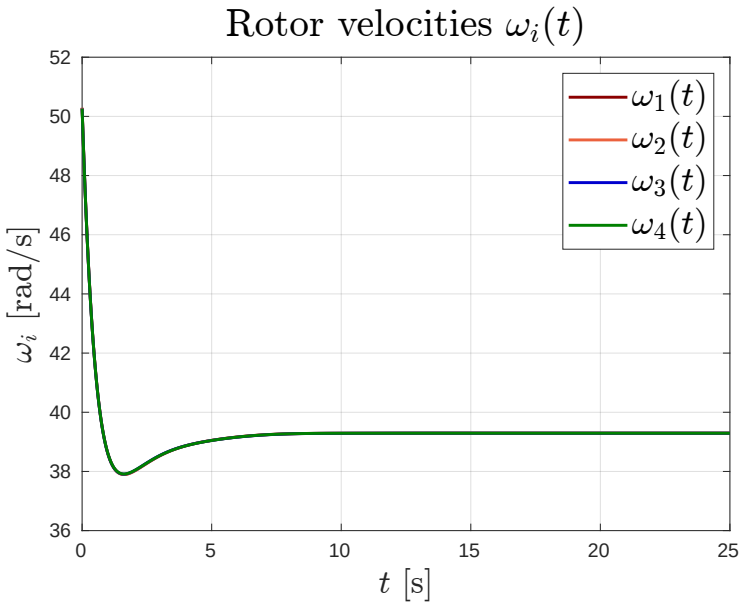

Figure 5.6 Rotor velocities over time as calculated using (4.38), the actuation required for trajectory tracking.

the PSO framework. In the following sections we report the results relative to 3 different scenarios regarding the number of drones employed and the positions of the victims in space, under nominal conditions both for the ARTVA signal and the PID controller.

5.2.3 Case 1

In the first case, the PSO algorithm needs to locate four sources positioned at $[20, 50]$ m, $[-50, 70]$ m, $[60, 30]$ m, and $[-10, 30]$ m, randomly distributed across the search space.

The time it took for the swarm to find the victims was 214 seconds (~ 3.6 minutes), during which the drones successfully converged on the sources, achieving precise estimates with an average error of 0.04 m. The estimate for the first source was just 0.125 m away, while the second, third, and fourth sources were located with errors of 0.037 m, 0.005 m, and 0.002 m, respectively.

In Figure 5.7 we can see the sources' locations and the drones' trajectories duing the PSO optimization process, from which is evident that the exclusion zone mechanism worked in favour of drone 3 in successfully locating a source.

5.2.4 Case 2

In the second case, the PSO algorithm is again tasked with locating four sources positioned at $[-70, -70]$ m, $[70, 70]$ m, $[-70, 70]$ m, and $[70, -70]$ m, distributed very far apart across the search space.

The time it took for the swarm to locate the sources was 227 seconds (~ 3.8 minutes). During this time, the drones successfully converged on the sources, achieving an average error of

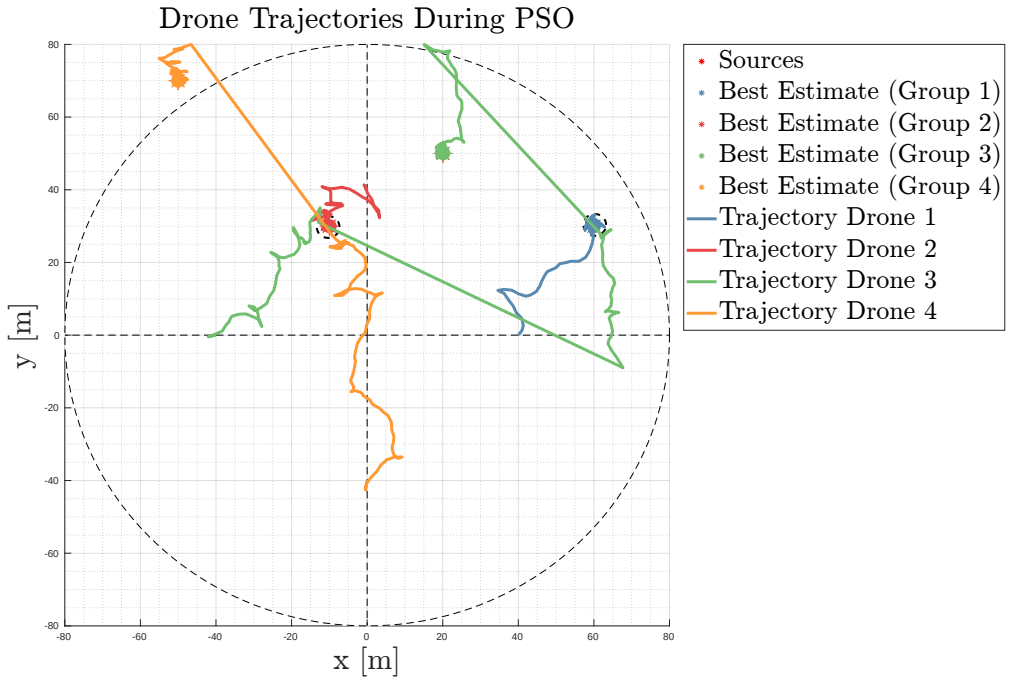

Figure 5.7 Results and Trajectories in Case 1.

0.26 m. The estimates for all the sources were: the first source was 0.549 m away, the second was 0.003 m away, the third was 0.478 m away, and the fourth was 0.012 m.

In Figure 5.8, the sources' locations and the drones' trajectories during the PSO optimization process are depicted. In this case, we can see how the trajectory mechanism was not necessary in guiding the drones towards new sources.

To evaluate the impact of increasing the number of drones, a second simulation was conducted by adding an additional drone to the swarm. Naturally, we see improved performance in the total time needed by the UAVs to find all victims, which was just 150 seconds ($\sim$ 2.5 minutes). The average error across all sources was reduced to 0.11 m. This improvement was due to drone 3 now having a different trajectory and initial position (after the exploration phase), which was nearer the source than before. Figure 5.9 illustrates the results of this improved configuration.

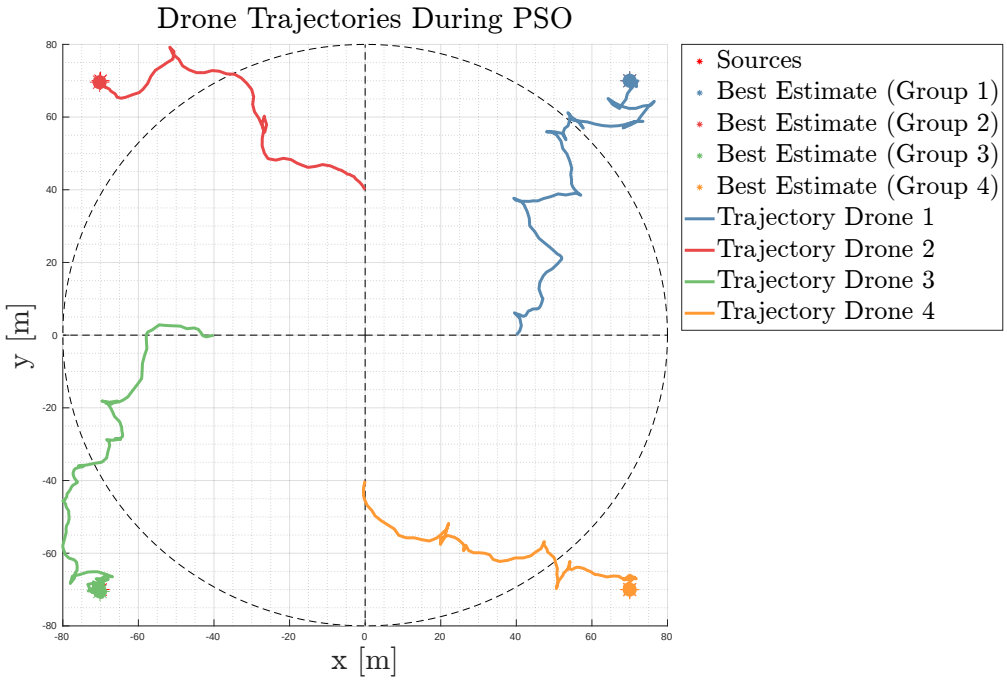

Figure 5.8 Results and Trajectories in Case 2.

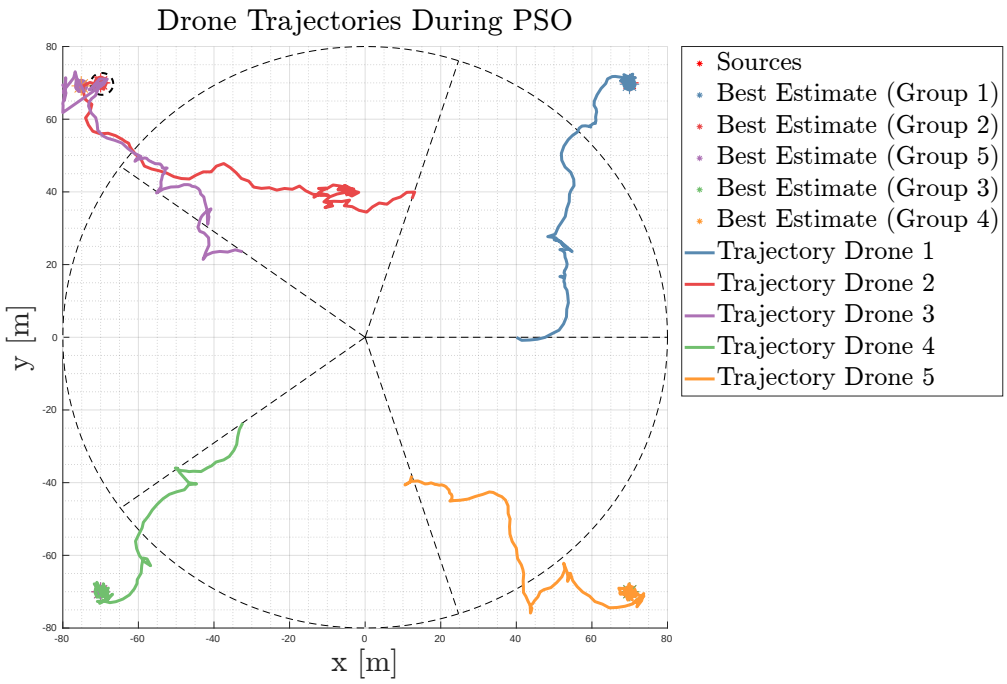

Figure 5.9 Results and Trajectories in Case 2 with an additional drone.

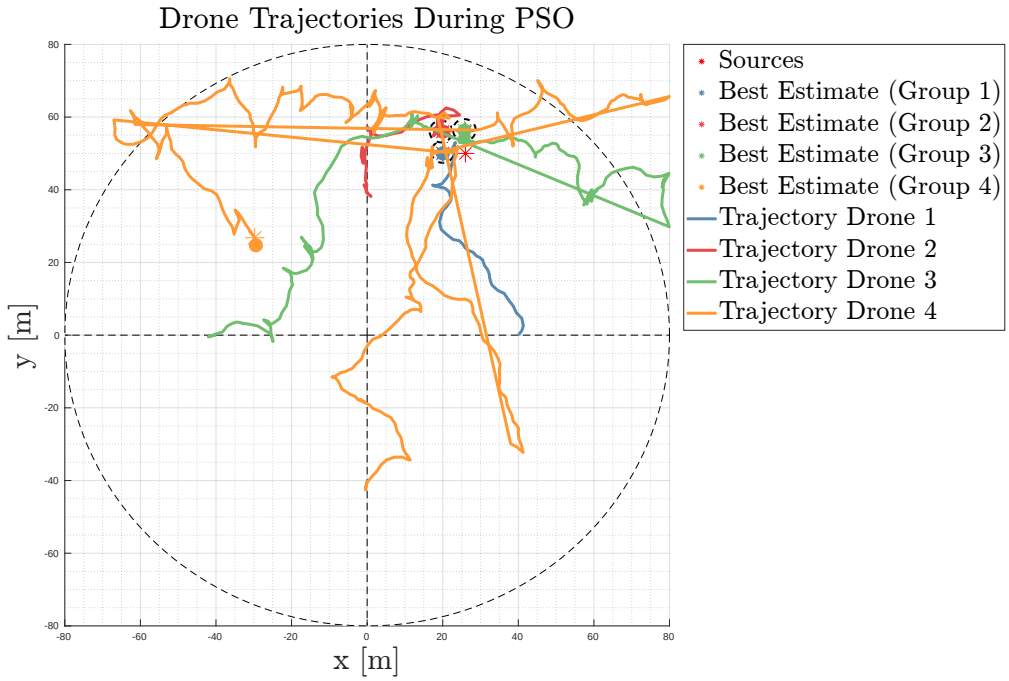

Figure 5.10 Results and Trajectories in Case 3 with fail.

5.2.5 Case 3

In the third case, the PSO algorithm faces a challenging scenario where the four sources are positioned very close to each other, only 6 m apart, at $[20, 50]$ m, $[26, 50]$ m, $[20, 56]$ m, and $[26, 56]$ m. This situation is particularly difficult for the algorithm due to the fact that we have an exclusion zone radius of 3 m and whenever one drone gets inside the exlusion zone of another it has to go 90 m away. Therefore, when three drones have found their sources, the fourth has only a few possible directions to reach the fourth source.

Initially, with four drones, the algorithm fails to reliably locate all sources within the 450 seconds allocated for the simulation, as seen in Figure 5.10. While increasing the simulation time could potentially improve performance, this approach is unsuitable for our time-sensitive application, where avalanche victims have a real chance of surviving only within 15 minutes after the time of the accident.

To address this issue, an additional drone is introduced, increasing the swarm size to five drones. Now, the algorithm successfully locates all sources in 369 seconds (approximately 6.15 minutes). The estimates for the sources are: the first source was 0.085 m away, the second was 0.022 m away, the third was 0.024 m away, and the fourth was 0.500 m away. The total error for all valid estimates was reduced to 0.16 m, demonstrating the effectiveness of increasing the swarm size in challenging scenarios. In Figure 5.11, the addition of a fifth drone and the subsequent success in the localization can bee seen clearly.

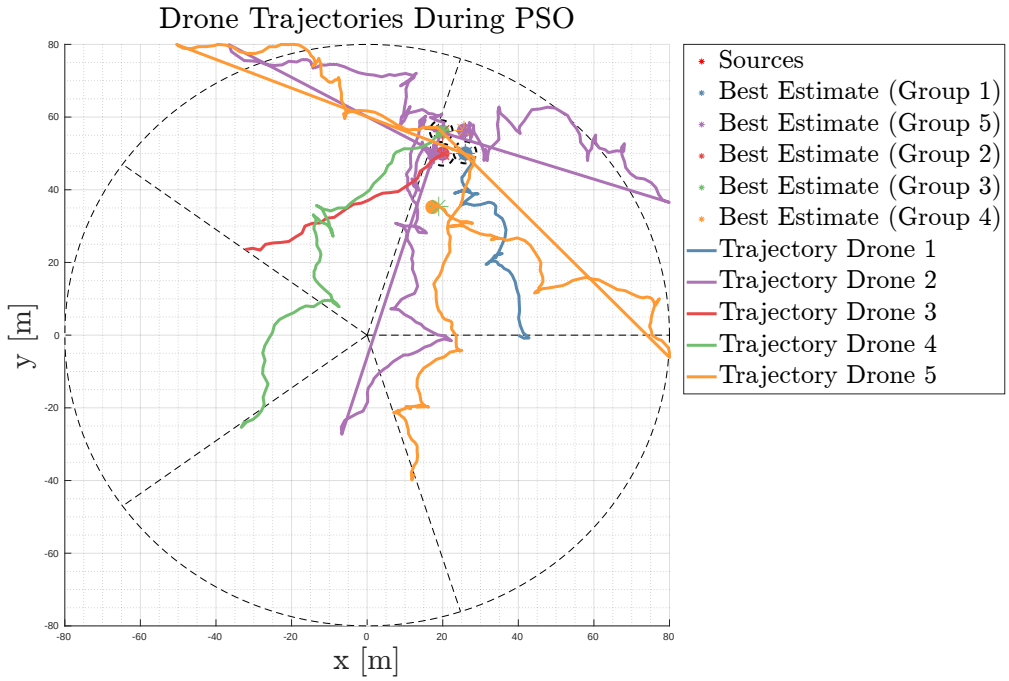

Figure 5.11 Results and UAVs Trajectories in Case 3 with an additional drone.

5.3 Conclusions

In this work, we modify the PSO algorithm to address the challenging task of multi-source localization using a decentralized swarm of UAVs. The results show that, in general, the algorithm produces reliable performance with precise estimates and fast convergence times, even in complex configurations. However, as an inherently heuristic approach, the PSO is not guaranteed to always converge, and certain situations highlight its potential drawbacks.

One of the main challenges we face is overcoming the superposition in signal strength from multiple sources or victims. This is our primary focus, as it inherently complicates the identification of individual sources. To address this issue, we implement a strategy (the PSO) that allows the swarm to localize multiple sources based on the aggregated signal, effectively bypassing the need to separate and identify individual ones. While this approach proves successful in enabling the swarm to locate sources in challenging scenarios, it introduces a trade-off: the certainty of finding a source is reduced because the PSO heuristic approach relies on some degree of randomness and the exclusion zone mechanism.

Another notable drawback is that the algorithm occasionally fails in scenarios where drones are repeatedly attracted to the same sources, preventing effective exploration of the search space. This limitation prevents the swarm's ability to identify and reach faraway sources, when near others. This behavior is an intrinsic limitation of the problem. Additionally, when sources are positioned too closely together, as demonstrated in Case 3, the algorithm struggles to differentiate between them, resulting in reduced accuracy unless additional drones are introduced to improve coverage.

One potential enhancement in our algorithm is the emplyment of an adaptive mechanisms for tuning the PSO parameters, such as inertia weight which could dynamically adjust based on the swarm's progress or environmental conditions, improving convergence speed and accuracy. In addition, investigating hybrid optimization methods that combine the PSO with local search or other heuristic approaches could provide a more comprehensive solution for complex multi-source localization tasks.

On the other hand, the algorithm's strengths lie in its adaptability and the benefits of a decentralized approach. The exclusion zone mechanism enables the swarm to dynamically adjust its search strategy based on the locations of previously detected sources, promoting faster convergence and improved accuracy. This adaptability is especially valuable in multi-source localization problems, where the location of all different sources rapdly is crucial. Furthermore, our work also introduces aother novel aspect: the implementation of the PSO algorithm to provide control command velocities to the drones in real time, integrating the PSO in the control loop.

The decentralized nature of our approach offers further advantages. By removing the reliance on a central computation unit, the system becomes more robust to communication noise and failures. This autonomy allows the drones to operate effectively under challenging conditions, such as mountainous terrain or adverse weather, where communication may be intermittent or delayed. While decentralization may lead to slightly slower convergence and higher estimation errors due to partial information exchange, it remains an effective solution in scenarios where centralized coordination is impractical or resource-intensive.

In conclusion, our implementation demonstrates that while the modified PSO algorithm encounters limitations in specific cases, it provides a powerful and flexible tool for decentralized multi-source localization. By addressing the challenge of signal overlap, our algorithm proves to be both effective and efficient in complex conditions and environment associated with avalanche victims locatization.

Bibliography

[1] L. Pedersen and T. Rasmussen, "The gradient tensor of potential field anomalies: Some implications on data collection and data processing of maps," *Geophysics*, vol. 55, pp. 1558–1566, Dec. 1990. DOI: 10.1190/1.1442807.

[2] M. Falk, H. Brugger, and L. Adler-Kastner, "Avalanche survival chances," *Nature*, vol. 368, no. 6466, pp. 21–21, 1994.

[3] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[4] E. Altug, J. Ostrowski, and R. Mahony, "Control of a quadrotor helicopter using visual feedback," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 1, 2002, 72–77 vol.1. DOI: 10.1109/ROBOT.2002.1013341.

[5] P. Schmidt, D. Clark, K. Leslie, M. Bick, D. Tilbrook, and C. Foley, "Getmag - a squid magnetic tensor gradiometer for mineral and oil exploration," *Exploration Geophysics - EXPLOR GEOPHYS*, vol. 35, Dec. 2004. DOI: 10.1071/EG04297.

[6] L. Derafa, T. Madani, and A. Benallegue, "Dynamic modelling and experimental identification of four rotors helicopter parameters," in *2006 IEEE International Conference on Industrial Technology*, 2006, pp. 1834–1839. DOI: 10.1109/ICIT.2006.372515.

[7] T. Madani and A. Benallegue, "Backstepping control for a quadrotor helicopter," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 3255–3260. DOI: 10.1109/IROS.2006.282433.

[8] T. Nara, S. Suzuki, and S. Ando, "A closed-form formula for magnetic dipole localization by measurement of its magnetic field and spatial gradients," *IEEE Transactions on Magnetics*, vol. 42, no. 10, pp. 3291–3293, 2006. DOI: 10.1109/TMAG.2006.879151.

[9] P. Pinies and J. Tardos, "Fast localization of avalanche victims using sum of gaussians," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, 2006, pp. 3989–3994. DOI: 10.1109/ROBOT.2006.1642314.

[10] P. Pinies, J. D. Tardos, and J. Neira, "Localization of avalanche victims using robocentric slam," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 3074–3079. DOI: 10.1109/IROS.2006.282247.

[11] R. W. Beard, "Quadrotor dynamics and control," 2008. [Online]. Available: https://api.semanticscholar.org/CorpusID:195351003.

[12] Y. Liu, W.-t. Zhou, Y. Liang, Q. Pan, and Y.-m. Cheng, "A novel pso based acoustic source localization algorithm in wireless sensor network," in *2010 8th World Congress on Intelligent Control and Automation*, 2010, pp. 820–825. DOI: 10.1109/WCICA.2010.5554104.

[13] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control* (Advanced Textbooks in Control and Signal Processing). Springer London, 2010, ISBN: 9781846286414. [Online]. Available: https://books.google.it/books?id=jPCAFmE-logC.

[14] S. Waharte and N. Trigoni, "Supporting search and rescue operations with uavs," in *2010 International Conference on Emerging Security Technologies*, 2010, pp. 142–147. DOI: 10.1109/EST.2010.31.

[15] M. Beiki, D. A. Clark, J. R. Austin, and C. A. Foss, "Estimating source location using normalized magnetic source strength calculated from magnetic gradient tensor data," *Geophysics*, vol. 77, no. 6, J23–J37, 2012, ISSN: 0016-8033. DOI: 10.1190/geo2011-0437.1. [Online]. Available: https://doi.org/10.1190/geo2011-0437.1.

[16] D. Clark, "New methods for interpretation of magnetic gradient tensor data," *ASEG Extended Abstracts*, vol. 2012, Dec. 2012. DOI: 10.1071/ASEG2012ab081.

[17] D. Clark, "New methods for interpretation of magnetic vector and gradient tensor data i: Eigenvector analysis and the normalised source strength," *Exploration Geophysics*, vol. 43, pp. 267–282, Sep. 2012. DOI: 10.1071/EG12020.

[18] T. Lee, M. Leok, and N. H. McClamroch, "Nonlinear robust tracking control of a quadrotor uav on se(3)," in *2012 American Control Conference (ACC)*, 2012, pp. 4649–4654. DOI: 10.1109/ACC.2012.6315143.

[19] 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. DOI: 10.1109/MRA.2012.2206474.

[20] L. Marconi *et al.*, "The sherpa project: Smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments," in *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2012, pp. 1–4. DOI: 10.1109/SSRR.2012.6523905.

[21] J. Marsden and A. Tromba, *Vector Calculus*. Macmillan Learning, 2012, ISBN: 9781429224048. [Online]. Available: https://books.google.it/books?id=pVbIygAACAAJ.

[22] T. Tomic *et al.*, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 46–56, 2012. DOI: 10.1109/MRA.2012.2206473.

[23] D. Cheng, *Field and Wave Electromagnetics* (Addison-Wesley series in electrical engineering). Pearson Education Limited, 2014, ISBN: 9781292026565.

[24] D. Griffiths, *Introduction to Electrodynamics*. Pearson Education, 2014, ISBN: 9780321972101. [Online]. Available: https://books.google.it/books?id=J9ygBwAAQBAJ.

[25] A. Lanzon, A. Freddi, and S. Longhi, "Flight control of a quadrotor vehicle subsequent to a rotor failure," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 2, pp. 580–591, 2014. DOI: 10.2514/1.59869. eprint: https://doi.org/10.2514/1.59869. [Online]. Available: https://doi.org/10.2514/1.59869.

[26] J. Zhang, D. Gong, and Y. Zhang, "A niching pso-based multi-robot cooperation method for localizing odor sources," *Neurocomputing*, vol. 123, pp. 308–317, 2014, Contains Special issue articles: Advances in Pattern Recognition Applications and Methods, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2013.07.025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231213007698.

[27] G. Bevacqua, J. Cacace, A. Finzi, and V. Lippiello, "Mixed-initiative planning and execution for multiple drones in search and rescue missions," *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 25, no. 1, pp. 315–323, Apr. 2015. DOI: 10.1609/icaps.v25i1.13700. [Online]. Available: https://ojs.aaai.org/index.php/ICAPS/article/view/13700.

[28] W.-D. Chang, "A modified particle swarm optimization with multiple subpopulations for multimodal function optimization problems," *Applied Soft Computing*, vol. 33, pp. 170–182, 2015, ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2015.04.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494615002161.

[29] V. Ferrara, "Technical survey about available technologies for detecting buried people under rubble or avalanches," *WIT Transactions on the Built Environment*, vol. 150, pp. 91–101, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:110028936.

[30] N. Lynn and P. N. Suganthan, "Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation," *Swarm and Evolutionary Computation*, vol. 24, pp. 11–24, 2015, ISSN: 2210-6502. DOI: https://doi.org/10.1016/j.swevo.2015.05.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210650215000401.

[31] R. Zou, V. Kalivarapu, E. Winer, J. Oliver, and S. Bhattacharya, "Particle swarm optimization-based source seeking," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 865–875, 2015. DOI: 10.1109/TASE.2015.2441746.

[32] J. Cacace, A. Finzi, V. Lippiello, M. Furci, N. Mimmo, and L. Marconi, "A control architecture for multiple drones operated via multimodal interaction in search & rescue mission," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2016, pp. 233–239. DOI: 10.1109/SSRR.2016.7784304.

[33] J. Cacace, A. Finzi, and V. Lippiello, "Implicit robot selection for human multi-robot interaction in search and rescue missions," in *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2016, pp. 803–808. DOI: 10.1109/ROMAN.2016.7745211.

[34] Y. Gang, Z. Yingtang, F. Hongbo, L. Zhining, and R. Guoquan, "Detection, localization and classification of multiple dipole-like magnetic sources using magnetic gradient tensor data," *Journal of Applied Geophysics*, vol. 128, pp. 131–139, 2016, ISSN: 0926-9851. DOI: https://doi.org/10.1016/j.jappgeo.2016.03.022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0926985116300714.

[35] R. Naldi, M. Furci, R. G. Sanfelice, and L. Marconi, "Robust global trajectory tracking for underactuated vtol aerial vehicles using inner-outer loop control paradigms," *IEEE Transactions on Automatic Control*, vol. 62, no. 1, pp. 97–112, 2017. DOI: 10.1109/TAC.2016.2557967.

[36] C. Sampedro, A. Rodriguez-Ramos, H. Bavle, A. Carrio, P. de la Puente, and P. Campoy, "A fully-autonomous aerial robot for search and rescue applications in indoor environments using learning-based techniques," *Journal of Intelligent & Robotic Systems*, vol. 95, pp. 601–627, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:115873208.

[37] C. Xu, Z. Yi, L. Meng, K. Huang, and J. Dai, "Detection technology of multi-magnetic source in spacecraft based on magnetic field gradient tensor," *IOP Conference Series: Earth and Environmental Science*, vol. 237, no. 3, p. 032 021, Feb. 2019. DOI: 10.1088/1755-1315/237/3/032021. [Online]. Available: https://dx.doi.org/10.1088/1755-1315/237/3/032021.

[38] S. Chang, Y. Lin, Y. R. Zheng, and X. Fu, "Simultaneous detection of multiple magnetic dipole sources," *IEEE Transactions on Magnetics*, vol. 56, no. 9, pp. 1–11, 2020. DOI: 10.1109/TMAG.2020.3011630.

[39] G. Yin, L. Zhang, H. Jiang, Z. Wei, and Y. Xie, "A closed-form formula for magnetic dipole localization by measurement of its magnetic field vector and magnetic gradient tensor," *Journal of Magnetism and Magnetic Materials*, vol. 499, p. 166 274, 2020, ISSN: 0304-8853. DOI: https://doi.org/10.1016/j.jmmm.2019.166274. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304885319324448.

[40] I. A. Azzollini, N. Mimmo, L. Gentilini, and L. Marconi, "Uav-based search and rescue in avalanches using ARVA: an extremum seeking approach," *CoRR*, vol. abs/2106.14514, 2021. arXiv: 2106.14514. [Online]. Available: https://arxiv.org/abs/2106.14514.

[41] D. Eidenbenz *et al.*, "Survival probability in avalanche victims with long burial (≥60 min): A retrospective study," *Resuscitation*, vol. 166, pp. 93–100, 2021, ISSN: 0300-9572. DOI: 10.1016/j.resuscitation.2021.05.030. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0300957221002148.

[42] *ETSI EN 300 718-1 European Standard*, v2.2.1, Available at https://www.etsi.org/deliver/etsi_en/300700_300799/30071801/02.02.01_60/en_30071801v020201p.pdf, ETSI, Jun. 2021.

[43] N. Mimmo, P. Bernard, and L. Marconi, "Avalanche victim search via robust observers," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 4, pp. 1450–1461, 2021. DOI: 10.1109/TCST.2020.3016665.

[44] C. Tabasso, N. Mimmo, V. Cichella, and L. Marconi, "Optimal motion planning for localization of avalanche victims by multiple uavs," *IEEE Control Systems Letters*, vol. 5, no. 6, pp. 2054–2059, 2021. DOI: 10.1109/LCSYS.2021.3049314.

[45] X. Ding, Y. Li, M. Luo, J. Chen, Z. Li, and H. Liu, "Estimating locations and moments of multiple dipole-like magnetic sources from magnetic gradient tensor data using differential evolution," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–13, 2022. DOI: 10.1109/TGRS.2021.3094057.

[46] N. Duong Thi Thuy, D. Nam Bui, M. Duong Phung, and H. Pham Duy, "Deployment of uavs for optimal multihop ad-hoc networks using particle swarm optimization and behavior-based control," in *2022 11th International Conference on Control, Automation and Information Sciences (ICCAIS)*, 2022, pp. 304–309. DOI: 10.1109/ICCAIS56082.2022.9990164.

[47] G. Liu, Y. Zhang, C. Wang, Q. Li, F. Li, and W. Liu, "A new magnetic target localization method based on two-point magnetic gradient tensor," *Remote Sensing*, vol. 14, no. 23, 2022, ISSN: 2072-4292. DOI: 10.3390/rs14236088. [Online]. Available: https://www.mdpi.com/2072-4292/14/23/6088.

[48] Z. Yu, Z. Si, X. Li, D. Wang, and H. Song, "A novel hybrid particle swarm optimization algorithm for path planning of uavs," *IEEE Internet of Things Journal*, vol. 9, no. 22, pp. 22 547–22 558, 2022. DOI: 10.1109/JIOT.2022.3182798.

[49] P. Rochford, *Transformation of covariance matrices for counter unmanned aircraft system applications*, Jul. 2023. DOI: 10.13140/RG.2.2.29019.54565.

[50] X. Yang, H. Li, and Y. Huang, "An adaptive dynamic multi-swarm particle swarm optimization with stagnation detection and spatial exclusion for solving continuous optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 123, p. 106 215, 2023, ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2023.106215. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197623003998.

[51] *ARVA-NeoBT Pro User Guide*, Available at https://beaconreviews.com/manuals/ARVA-Neo_BT_Pro_2024-02-07.pdf, ARVA NIC-IMPEX, Feb. 2024.

[52] G. Liu, Y. Zhang, C. Wang, Q. Li, and W. Liu, "Novel magnetic dipole localization method based on normalized source strength," *IEEE Sensors Journal*, vol. 24, no. 16, pp. 26 159–26 170, 2024. DOI: 10.1109/JSEN.2024.3423342.

[53] W. Lv, P. Huang, Y. Yang, Q. Luo, S. Xie, and C. Fu, "A novel method of magnetic sources edge detection based on gradient tensor," *Minerals*, vol. 14, no. 7, 2024, ISSN: 2075-163X. DOI: 10.3390/min14070657. [Online]. Available: https://www.mdpi.com/2075-163X/14/7/657.

Ringraziamenti

SCRIVI QUI I RINGRAZIAMENTI