Underwater robots for karst and marine exploration : A study ofredundant AUVs

Huu-Tho Dang

THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITÉ DE MONTPELLIER

En Systèmes Automatiques et Microélectroniques (SYAM)

École doctorale : Information, Structures, Systèmes (I2S)

Unité de recherche : LIRMM (UMR 5506)

UNDERWATER ROBOTS FOR KARST AND MARINE EXPLORATION: A STUDY OF REDUNDANT AUVs

Présentée par Huu Tho DANG

Le 26/05/2021

Sous la direction de Lionel LAPIERRE

Devant le jury composé de

René ZAPATA, Professeur, LIRMM, Montpellier	Président du jury
Lionel LAPIERRE, Maitre de Conférence, HDR , LIRMM, Montpellier	Directeur de thèse
Luc JAULIN, Professeur, Lab-STICC, Bretagne	Rapporteur
Vincent HUGEL, Professeur, COSMER, Toulon	Rapporteur
Antonio PASCOAL, Professeur, IST-ID, Portugal	Examinateur
Massimo CACCIA, Professeur, CNR-ISSIA, Italy	Examinateur
Olivier PARODI, IR, Naval Group	Examinateur
Benoit ROPARS, IR, Reeds	Examinateur

UNIVERSITÉ DE MONTPELLIER

Acknowledgments

First and foremost, I would like to thank my supervisor A. Prof Lionel Lapierre who recruited me more than 3 years ago, introduced me underwater domain and karst exploration specifically. Thanks for his guidance in underwater robotic field which was quite new for me in the first time. He not only helps me in sciences but also in my life in France. Normally, I gained a lot of things after discussing with him with very interesting questions. I also express my graceful to Prof René Zapata who always has new ideas in every problem. Personally, he is always a motivation of young researchers because of his hard working and interesting ideas.

I would like to express my gratitude to Prof Didier Crestani for his administrative works. A. Prof Sebastien Druon, A. Prof Karen Godary-Dejean, A. Prof Jean Triboulet for their open-mind.

I would like to thank Dr Benoit Ropars, Mr Pascal Lepinay, Mr Eric Dubreuil of Polytech, Montpellier, for their helps in printing and creating pieces of robot, carrying out experiments.

I would like to thank other PhD students of Explore team : Andrien Hereau, Quentin Massone, Rodolfo Villalobos Martinez, Philippe Lambert, and Dr Yohan Breux for discussions and coffee time.

I would like to thank my Vietnamese friends in Montpellier for their helps in my daily life and football matches at weekend in which I can release stress and get more energy for a new week. Especially, I would like thank my closed friend, Mr Nguyen Tuan Hung, from IST (Instituto Superior Técnico), Portugal, for his fruitful discussions in underwater robotics.

My sincere appreciation goes to the committee of my thesis, Prof René Zapata, Prof Luc Jaulin, Prof Vincent Hugel, Prof Antonio Pascoal, Prof Massimo Caccia, Dr Olivier Parodi, and Dr Benoit Ropars for their acceptance and advices.

And last but not least, I am indebted to my wife, my son, and my small daughter who always beside me during time in France, my parents, and my younger brother who always encourage me during PhD journey.

Without these helps, I can not finish this thesis.

I would like to say many thanks to the Labex NUMEV, Montpellier University, the Region Occitanie and the FEDER funds who sponsored for this work.

Underwater robots for karst and marine exploration : A study of redundant AUVs

Abstract : Nowadays, underwater environments including ocean and karst system with their biodiversity are needed to discover and underwater vehicles are means to do that. An underwater vehicle can normally be able to classify into an under-actuated system, a fully-actuated system, or an over-actuated one, also called redundant system. This depends on the number of actuators and the number of controllable Degrees of Freedom (DoFs). This property decides other relating problems such as controller design, kinds of missions, building cost. Although a redundant system has drawbacks, i.e., high building cost, not easy to control, it also has a lot of merits : a good choice for fault-tolerance control, be more flexible in operations. Moreover, with redundancy, it is obvious to vary configuration in order to inherit advantages of each kind of configurations. For application speaking, underwater vehicles have a wide range of employment from civil to military such as seabed investigation, biological discovery, marine rescue, and coast monitoring. Among of them, karst exploration is a potential domain and needs to study more because of the diversity of interior ecological systems and vital role in supplying and maintaining sweet water resource, especially in European countries.

Motivated by redundant feature and karst exploration, this work focuses on redundant systems in underwater robot field for exploring karst networks. Underwater vehicles with static configuration and reconfigurable one were studied. In particular, a method was proposed to determine the static configuration of a robot (position and direction of thrusters) in order to optimize performance indices : manipulability, workspace, energy, reactive, and robustness indices. An optimal solution based on multiobjective optimization was found. Simulation and experimental results were shown to prove the proposed approach. Considering advantages of a dynamically reconfigurable robot, a prototype robot, called Umbrella Robot, was built. A locally optimal approach with respect to energy-like criterion with dynamic configuration was suggested. In the meanwhile, control allocation methods for this kind of systems were investigated. Simulation results for two cases, simple guidance in which desired control vector is given and complex one in which desired control vector is able to vary in each time step, were illustrated. Our simulation and experiments validated the proposed method.

Keywords : Underwater robot, configuration matrix, multiobjective optimization, optimization.

Robots sous-marins pour l'exploration karstique et marine : une étude des AUVs redondants

Abstract : De nos jours, les environnements aquatiques, considérant l'océan et les systèmes karstiques noyés sont le lieu d'enjeux sociétaux majeurs, depuis la question des ressources (alimentaires et en eau) jusqu'à la compréhension des phénomènes physiques qu'ils hébergent. La difficulté d'accès à ces environnements fait du robot sous-marin un outil d'exploration efficace.

Les propriétés d'actionnement de ces systèmes robotisés spécifient leur capacité dynamique et les rendent adaptés à certains types de mission. Ils peuvent être sous-actionnés, iso-actionnés ou redondants. Cela dépend du nombre d'actionneurs et du nombre de Degrés de Liberté contrôlables.

Cette propriété de l'actionnement conditionne la stratégie de commande du système, engendre une certaine complexité au système mais comporte aussi des avantages : robustesse, tolérance aux pannes, pilotage de la réactivité et optimisation de la consommation énergétique globale. De plus, un système à géométrie d'actionnement variable permet de couvrir l'ensemble de ces propriétés, rendant le système versatile et configurable en fonction des contraintes environnementales et de mission. A l'instar des applications civiles et militaires, ce type de système présente des avantages majeurs pour la question de l'exploration karstique. En effet la variabilité des conditions environnementales de ce milieu confiné requiert une adaptabilité du système d'actionnement et un pilotage fin de sa réactivité. L'exploitation des propriétés de redondance du système d'actionnement, qu'il soit statique ou dynamiquement reconfigurable, est le sujet de cette thèse. En particulier, une méthode est proposée pour déterminer la configuration statique d'un robot (position et direction des propulseurs) afin d'optimiser les indices de performance : manipulabilité, atteignabilité, énergie, réactivité et robustesse. Une solution basée sur l'optimisation multi-objectifs est proposée. Cette approche est validée en simulation et expérimentalement sur 2 types de robots : le robot 'Cube' (configuration redondante statique) et le robot 'Umbrella' (redondant à géométrie variable). Une approche localement optimale, en ce qui concerne un critère de type énergie, permet une adaptation dynamique de la configuration d'actionnement. Une solution de répartition des actions individuelle des moteurs est aussi proposée. Les performances des solutions proposées sont illustrées en détail en simulation et une validation expérimentale a été effectuée

Keywords : robot sous-marin, matrice de configuration, optimisation.

Résumé de la thèse

Robots sous-marins pour l'exploration karstique et marine : une étude des AUVs redondants

Ces travaux de thèse ont profité d'un financement issu du Labex Numev (projet étendard Aleyin) et de la Région Occitanie.

1. Contexte

L'exploration océanique, en général, et l'exploration d'un environnement confiné tel que le karst pose des défis scientifiques et technologiques difficiles. Les véhicules autonomes ou semi-autonomes sous-marins sont développés pour répondre à de tels défis. Un robot sous-marin peut être catégoriser en fonction des propriétés de son actionnement. Il peut être :

1. Sous actionné, si son actionnement ne permet pas d'agir sur l'intégralité des (6) degrés de libertés.
2. Iso-actionné, s'il peut agir sur tous les degrés de liberté.

La configuration de ses moteurs peut aussi le rendre redondant, s'il existe plusieurs manières de réaliser un effort désiré. Ainsi un robot sous-marin peut être à la fois sous-actionné et redondant. Dans ces travaux nous nous intéressons à des systèmes redondants, dont la géométrie variable leur confère la possibilité d'évoluer d'un système iso-actionné à un système sous-actionné.

La gestion de la redondance d'actionnement peut être exploitée pour améliorer les performances du robot, telles que :

1. la tolérance aux pannes.
2. la réactivité du système.
3. la propriété d'isotropie de l'actionnement.
4. la minimisation de la dépense énergétique.
5. l'optimisation de la génération d'une force maximale dans une direction donnée.
6. La gestion des non-linéarités (zones mortes) des propulseurs.

Le contexte applicatif particulier de ces travaux de recherche impose de considérer les particularités de l'environnement confiné dans lequel le robot évolue. En effet le contexte karstique, décrit ci-après, impose de traiter de la question de l'allocation de contrôle avec soin.

Le karst, en général, consiste en un réseau de conduits naturels souterrains, résultant de la dissolution de roches solubles, telles que le calcaire, la dolomie, le gypse... Il est le lieu de la ressource en eau souterraine et, à ce titre, joue un rôle fondamental dans l'approvisionnement en eau potable de millions de personnes dans

le monde. Il peut contenir d'énormes quantités d'eau et la dynamique des transferts de charge peut engendrer des mouvements d'eau imprévisibles et provoquer des inondations soudaines et dramatiques, en particulier dans notre région sujette aux épisodes cévenols.

Ainsi, la question de la modélisation de la dynamique de ces réseaux karstique est un enjeu important qui requiert l'acquisition de données de terrain difficilement accessible par un humain. Ainsi la solution robotique semble être adaptée à ce contexte difficile, dans la mesure où les verrous scientifiques et technologiques propres à cet environnement confiné auront été résolus. Les robots permettront d'acquérir une information fiable, avec des protocoles répétables pour effectuer des missions de cartographie et de prélèvement saisonniers de manière à fournir aux autorités des données éclairant la prise de décision dans le cadre d'une gestion active de la ressource.

Historique des initiatives d'exploration karstique au moyen de robots

Les premières explorations karstiques robotisées ont eu lieu dans la légendaire Fontaine de Vaucluse et les retours d'expérience sont riches d'enseignements. Le premier robot d'exploration karstique en France est le Télénaute, en 1967, exploité par l'IFP (Institut Français du Pétrole). Il atteint la profondeur de 106m. En 1983, une nouvelle génération de robot voit alors le jour avec le Sorgonaute, construit en collaboration avec Renault. Il atteint les 243 m de profondeur et est arrêté par la longueur de son câble. Sorgaunote II est alors développé, mais est perdu à 233m, les moteurs de celui-ci s'étant pris dans une ligne de vie de plongeur. En 1986, Sorgaunaute III est construit avec pour mission principale la récupération de son prédécesseur. Il est perdu à 150m, s'étant emmêlé dans le câble de Sorgaunaute II. Enfin en 1988, Sorgaunote IV est envoyé, mais échoue lui aussi. Des plongeurs scaphandriers sont finalement envoyés pour désobstruer la source. En 1985, le Modexa atteint le fond du gouffre, à 315m. D'autres initiatives sont rapportées dans la littérature (DEPTHX, UNEXMIN...), et toutes soulignent la difficulté principale de ce type d'opération : la gestion du câble ombilical qui vient se prendre dans les reliefs de l'environnement et cause la perte des engins.

La stratégie du projet Aleyin

L'équipe EXPLORE du LIRMM, Université de Montpellier, porte le projet ALEYIN qui propose d'aborder la question de l'exploration karstique avec la stratégie suivante :

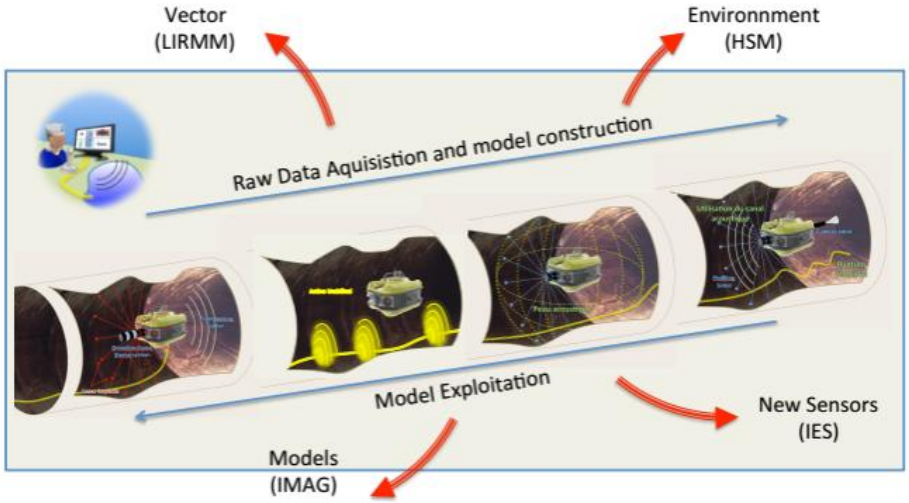


FIGURE 1 – Stratégie retenue pour l'exploration karsrtique dans le cadre du projet Aleyin (Labex NUMEV)

Deux phases sont distinguées (cf. Figure 1) :

1. La phase d'*exploration*, durant laquelle le robot est téléopéré et muni d'un trancaneur qui lui permet de déposer sont câble durant la progression. Ses capteurs acoustiques lui permettent d'acquérir des mesures sur la géomorphologie du terrain et la mission est dirigée par l'expert hydrogéologue qui profite des données acquises par le robot en temps réel.
2. La phase de *retour* (homing), durant laquelle le robot est libéré de son câble et exploite la carte construite à l'aller (SLAM) pour naviguer en toute autonomie.

Parmi l'ensemble des défis scientifiques et technologiques que soulève cette stratégie (capteurs, communication, navigation, SLAM, autonomie...), cette thèse aborde la question fondamentale de celle de l'actionnement, à savoir : que doit être une *bonne* configuration de l'actionnement d'un robot pour réaliser de manière fiable des opérations d'exploration karstique.

Du point de vue du contrôle, la géométrie d'actionnement est exprimée sous la forme d'une *matrice de configuration* qui est construite à partir de la position et de la direction des actionneurs. L'étude des propriétés algébriques de cette matrice permet de contraindre les stratégies de contrôle, mais détermine aussi un ensemble de qualités (réactivité, manœuvrabilité, robustesse aux pannes, compensation des non-linéarité des propulseurs, puissance propulsive, réduction de la consommation énergétique...) qui affectent le système global. Deux systèmes sont conçus, réalisés et expérimentés dans le cadre de cette thèse : le robot redondant statique *Cube* et le robot à géométrie variable *Umbrella* (cf Figure 2).

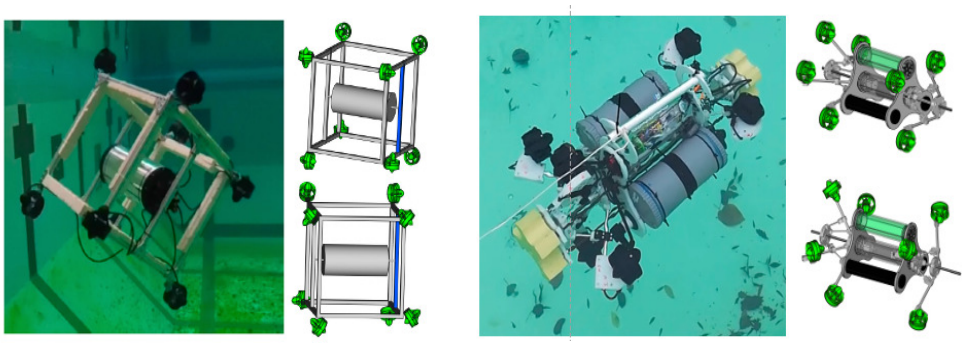


FIGURE 2 – Le robot *Cube* et ses deux configurations statiques (gauche). Le robot *Umbrella* et ses configurations dynamiques.

2. Organisation et contenu du manuscrit

Le manuscrit se compose de neuf chapitres.

Chapitre 1

Il présente les rôles importants que les systèmes karstique et océaniques et la nécessité de leur exploration. Une vue d'ensemble et une classification des robots sous-marins existants est proposée et un historique rapide de l'utilisation de ces robots pour l'exploration karstique est présentée. Les enjeux robotiques spécifiques à l'exploration karstique sont présentés.

L'architecture et la configuration des robots sous-marins autonomes sont également discutées. Enfin, la motivation et les contributions de la thèse sont établies et la structure de la thèse est présentée.

Chapitre 2

Ce chapitre aborde les fondamentaux de la robotique sous-marine, des modèles cinématiques aux modèles dynamiques, les différents formalismes de représentation des orientations, les stratégies de contrôle non linéaire et l'allocation de contrôle sont brièvement présentés. Plus précisément, le modèle cinématique des véhicules sous-marins peut être présenté dans le formalisme d'Euler ou quaternion dans lequel le blocage du cardan peut être évité. Certaines des méthodes de contrôle non linéaires : programmation de gain, linéarisation par rétroaction, *backstepping* et contrôle prédictif sont résumés. Une revue de la littérature sur les méthodes d'allocation de contrôle, qui jouent un rôle essentiel dans le contrôle des systèmes redondants, est présentée. Les différentes méthodes d'allocation de contrôle sont : la méthode directe, la méthode en guirlande, la cascade générale inverse (CGI) et les méthodes de programmation linéaire/non linéaire. Enfin, la fonction d'erreur du contrôle, basée sur le formalisme des quaternions, qui est un guide pour le contrôleur dans le chapitre suivant, est présentée et démontrée.

Chapitre 3

Les techniques différentes d'optimisation et d'optimisation multi-objectifs sont présentées dans ce chapitre. Ce sont les outils de base pour résoudre les problèmes proposés dans les chapitres suivants. Les principes fondamentaux d'optimisation sont présentés, tels que les conditions de Karush-Kuhn-Tucker (KKT) et le multiplicateur de Lagrange. La programmation quadratique séquentielle pour un problème nonlinéaire avec la méthode des ensembles actifs, qui est partiellement utilisée dans le chapitre 7 est exposée et discutée. Pour l'optimisation multi-objectif, les définitions de la solution de Pareto et du front de Pareto sont présentées. Les différentes variantes des méthodes de résolutions sont présentées : pondération, ε-contrainte, critère global, ordre lexicographique, etc. Pour les méthodes heuristiques, différents algorithmes sont considérés : Algorithme génétique multiobjectif, algorithme génétique non dominé, optimisation d'essaim partiel multi-objectif, algorithme évolutif multi-objectif... Les avantages et les inconvénients de chaque méthode sont discutés.

Chapitre 4

Ce chapitre étudie les propriétés d'une configuration statique pour les robots sous-marins redondants et différents indices de performance sont proposés et analysés : manipulabilité, optimisation énergétique, espace de travail et atteignabilité, réactivité et robustesse.

Plus précisément, l'indice de manipulabilité exprime les propriétés isotropes du robot. Cela signifie que le robot peut agir de la même manière dans les 6 degrés de liberté. L'indice énergétique exprime la consommation d'énergie lorsque le robot doit exercer un effort désiré dans une direction donnée. L'indice relatif à l'espace de travail montre la possibilité d'atteindre le vecteur de contrôle souhaité. L'indice de réactivité indique à quelle vitesse le robot peut modifier l'orientation et l'amplitude de l'effort qu'il exerce sur l'environnement. Enfin, l'indice de robustesse établit si le robot peut continuer à agir selon les 6 degrés de liberté lorsqu'un ou plusieurs moteurs sont complètement défaillants. L'analyse mathématique de chaque critère est illustrée. Le problème est formulé comme un problème d'optimisation multi-objectif. Une méthode d'optimisation multi-objectifs, appelée *objectif atteint*, est choisie et une procédure de recherche d'une solution optimale est proposée. Cette solution est une solution optimale appartenant à la frontière de Pareto. La classification de l'ensemble des solutions de la frontière est encore un sujet ouvert et sera l'objet d'une étude future.

Chapitre 5

Des simulations et des résultats expérimentaux de différentes conception de configuration statique, dans lesquels la matrice de configuration est constante, sont présentés. Une comparaison entre deux configurations est effectuée pour illustrer les performances de l'approche proposée. En particulier, des simulations réalisées avec deux configurations sont discutées. Le premier est pour le cas général dans lequel

la position et l'orientation des propulseurs sont inconnues. Le second pour le cas spécifique dans lequel la position des propulseurs est contraintes sous la forme de cube.

La conception et la réalisation du robot *Cube* est présentée et deux configurations sont testées en simulation et expérimentalement. L'une est appelée configuration normale dans laquelle les propulseurs sont fixés verticalement ou horizontalement, suivant les arrêtes du cube. La seconde est la configuration optimale (une solution du chapitre 4). Tous les indices de performance sont établis, comparés et discutés. On vérifie ainsi expérimentalement que la configuration optimale donne de meilleures performances que la configuration normale.

Chapitre 6

Un nouveau prototype de robot sous-marin à géométrie variable, appelé robot *Umbrella*, est conçu et construit. La configuration de l'actionnement du robot dépend de deux angles (α_F et α_B) et peut être modifiée avec deux moteurs à courant continu supplémentaires. Cet actionnement supplémentaire permet de générer un système avec de bonnes propriétés de réactivité et de manœuvrabilité en mesure de maintenir une position dans l'espace (*station keeping*). La modification de sa configuration d'actionnement permet alors d'optimiser sa capacité propulsive (atteignaibilité dans une direction donnée), perdant sa capacité de se maintenir à l'arrêt, mais pouvant exercer l'effort maximal dans la direction principale de son avancement.

Des critères de capacité d'action le long des 6 degrés de liberté sont proposés et une optimisation géométrique supplémentaire est effectuée, suggérant des modifications du robot qui seront effectuée plus tard. Une analyse comparative des différents types de robots redondant en fonction des propriétés de leur système d'actionnement est effectuée.

Chapitre 7

La question de la configuration dynamique d'un robot est présenté dans ce chapitre. Un algorithme, appelé A-SQP, basé sur la programmation quadratique séquentielle mentionnée au chapitre 3, est proposé pour résoudre le problème. En particulier, dans ce chapitre, le robot peut changer sa configuration à chaque instant d'échantillonnage. Une fonction objectif de type énergie est proposée et un problème d'optimisation sous contraintes est formulé. Un algorithme est proposé pour résoudre le problème. De plus, des méthodes d'allocation de contrôle avec une configuration dynamique sont également étudiées et comparées. Les résultats montrent que dans certains cas, les méthodes basées sur la pseudo-inverse sont moins performantes que les méthodes basées sur la programmation nonlinéaire.

Chapitre 8

Les résultats de simulations et des expériences pour le robot Umbrella sont présentés dans ce chapitre. Pour la configuration dynamique, les résultats de simulation sont étudiés en fonction de différentes problématiques, à savoir les missions données, le suivi de trajectoire et le cas d'observation (maintien d'une position dans l'espace). Des simulations et des validations expérimentales de la capacité reconfigurable du robot Umbrella sont présentées. Il s'agit en particulier du contrôle de la tenue de la profondeur, de la régulation de l'attitude et d'une mission intégrée. Avec la configuration dynamique, différents cas sont simulés. Premièrement, le vecteur de contrôle souhaité est donné. Deuxièmement, le vecteur de contrôle désiré est issu du calcul du contrôleur. Ceci est simulé avec des problèmes de suivi de trajectoire et de maintien d'une position (*station keeping*). Les résultats de simulations montrent que l'algorithme proposé est efficace.

Chapitre 9

Les conclusions et de ces travaux futurs sont exposées. Les avantages et les limitations de la thèse sont présentés. Les travaux futurs et étendus sont également proposés. Les travaux futurs intéressants sont l'étude du front de Pareto du problème de configuration statique, l'application de la programmation paramétrique pour trouver un ensemble réalisable de vecteurs de contrôle désirés pour le problème de configuration dynamique, la mise en œuvre d'expériences réelles pour le problème de configuration dynamique, et enfin un nouveau mécanisme, hybride entre le système *Cube* et le robot *Umbrella* est proposé.

Contents

Nomenclature

$\mathbf{A}$	Configuration matrix
$\mathbf{A}^+$	Moore-Penrose pseudo-inverse of $\mathbf{A}$ matrix
$Ker(\mathbf{A})$	Kernel space of $\mathbf{A}$ matrix
$\mathbf{u}_i$	(unit) vector of direction of the i^{th} thruster w.r.t Body frame
$\mathbf{r}_i$	(unit) vector of position of the i^{th} thruster w.r.t Body frame
$F_{m,i}$	Force magnitude of the i^{th} thruster
$\mathbf{F}_m$	Force vector of m thrusters
$\mathbf{F}_m^d$	Desired force vector of m thrusters
$\mathbf{F} = (F_u \quad F_v \quad F_w)^T$	the vector of force elements in the resulting force $\mathbf{F}_B$
$\mathbf{\Gamma} = (F_p \quad F_q \quad F_r)^T$	the vector of torque elements in the resulting force $\mathbf{F}_B$
$\mathbf{F}_B = \binom{\mathbf{F}}{\mathbf{\Gamma}}$	Resulting force vector (force and torque elements) w.r.t Body frame
$\mathbf{F}_B^d$	Desired control vector (force and torque elements) w.r.t Body frame
$\mathbf{c}_m$	Input vector of thrusters (PWM)
$\otimes$	Cross product
$\odot$	Quaternion product
$\| \, . \, \|$	Euclidian norm
$\| \, . \, \|_p$	p-norm

List of Figures

List of Tables

Chapter 1

Introduction

Contents

This chapter presents important roles of karst system and ocean in human life and the needs of their exploration. The overview of underwater robot domain is also investigated with classification of robots and a rapid history of using robot for karst exploration is presented. The ideas and challenging questions for karst exploration are discussed. Finally, motivation and contributions of the thesis are also established.

1.1 Overview

As far as we know, water is an essential element to sustain life. Only 0.023% of the mass of the Earth is composed with water and is distributed according to Figure 1.1. The most part of this water is salted (96.5%) and forms the Global Ocean. Only 2.5% of this water is fresh and proper to be used in industry or for human consumption. However, only 1.2% of this fresh water is accessible at surface and, as a dramatic consequence, is now highly polluted, 68.7% is captured in glaciers and ice caps and 30.1% is present underground. This underground water, that represents 0.76% of the global resource, is of great value. Groundwater is the result of a natural and long filtration process through the different layers of the soil, and its quality is generally considered as very good. However, the difficulties to exploit this resource are important, and due to the lack of knowledge about the location and dynamic of the underground drainage system.

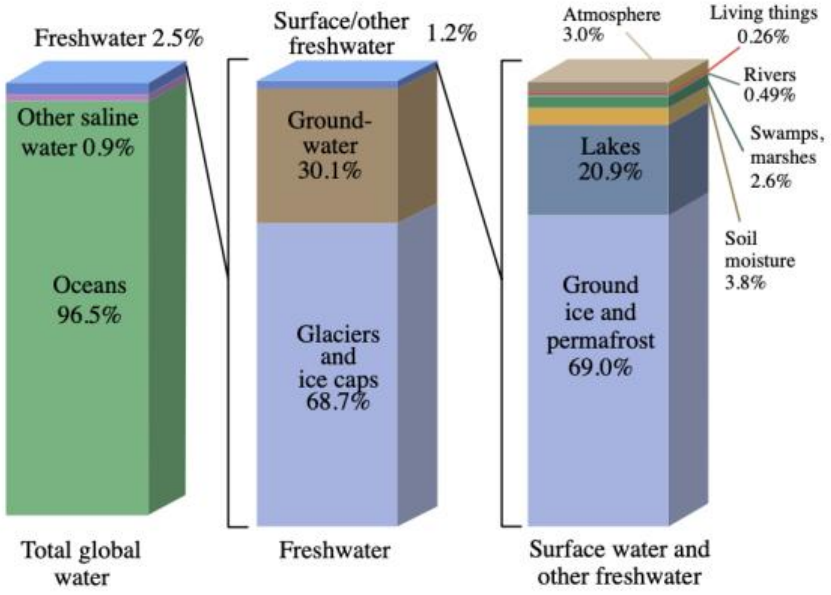


Figure 1.1 – Where is Earth's Water?, from [Gleick 1993]

Karst is a landscape formed by dissolving soluble rocks such as limestone, dolomite, and gypsum. It consists of sinkholes, caves, and underground drainage systems. The dissolution process has created a complex underground water flow network where rainwater can be stored. Many drinking water supplies for human-beings are extracted from karst aquifer. Moreover, more than 50% of the world's hydrocarbon reserves are hold in porous karst systems [Ford 2007a]. Besides, karst rocks, specifically carbonate rocks, are main elements of agricultural lime, Portland cement, fine building stones, and aggregate for highways [Ford 2007b]. Therefore, understanding karst systems not only plays important role in groundwater management but also in keeping biodiversity and maintaining karst structure.

The need to discover ocean, karst system, and underwater is a mysterious story of human-beings, from curious circumstance to economic issues. In fact, living and non-living resources of ocean are important parts in our life from food suppliers to heavy industries, i.e, fishing, mining, oil and gas. Moreover, nowadays, global warming phenomenon causes tsunami and salinization which thread millions of people all over the world, especially in countries bordering by oceans. Necessities of understanding ocean and sweet water dynamics are more essential than ever. Underwater Robots (URs) are powerful tools to help our to extend our knowledge of these subaquatic environments. To this end, studies and works in underwater robot field have been sharply developed in recent years. Many projects have been

invested to enhance capabilities of underwater robots in America and European countries [Huet 2016] [Zereik 2018]. A large amount of underwater robot applications have been seen in all domains from civil to military sections. A primary survey of underwater robot employments can be referred to [Lapierre 2006a].

1.2 Underwater robotics

Going back in history, the development of underwater robot has spent for a long time. Diving bell based on a design of *Leonardo da Vinci* in 1531 is considered as the first model of submarine; then the SPURV (Self-Propelled Underwater Research Vehicle, USA) and the Epaulard (France) are considered to be the first autonomous underwater vehicles models which were developed during the 60's and 70's [Lapierre 2006a]. The outbreak of Autonomous Underwater Robots (AUVs) happened in 90's with more than 46 models for various missions [Yuh 2000]. Up to day, there exists numerous of underwater robots that have been built not only for commercial market but also for scientific community. Regarding appearances and functions, underwater robots can be categorized as follows:

1. **Unmanned Surface Vehicles (USVs)**: USVs are deployed on the water surface and it is easy to receive *Global Position System* (GPS) signal for localization and navigation. Moreover, since their domain of evolution is 2-Dimension (2D), the control question may appear simpler. From an actuation point of view, they can optimize propulsion by utilizing wind or wave energy. However, environment disturbances (wind, waves and current) and boat traffic impact these surface systems and raise important scientific and technological challenges.
2. **Remotely Operated Vehicles (ROVs)**: ROVs can operate underwater and communicate with an operator by umbilical cable which allows for a real-time connection. Indeed, power supply, data and control commands may be transferred through this link. The first ROVs models came from the needs of oil and gas industry for deep sea missions. With expansion of underwater robot applications and forces in science, numerous smaller ROV models have been built for shallow water. However, ROVs are also limited in operating space with the length of umbilical and cumbersome link.
3. **Autonomous Underwater Vehicles (AUVs)**: Autonomous Underwater Vehicles are different from ROVs since they do not have umbilical link. AUVs can operate autonomously and, as a consequence, perform a smaller number of typical missions. AUVs are often divided into two categories: torpedo-shaped or cubic-shaped AUV (based on its appearance and dynamics). Torpedo-shaped AUVs are usually under-actuated systems and designed for long range missions. Otherwise, cubic-shaped AUVs are often fully actuated or over-actuated systems and useful for short range missions. Some different properties between two these kinds of AUVs are hovering and pivot steering capabilities or operating speed range as well.

4. **Intervention Autonomous Underwater Vehicles (IAUVs)**: Intervention Autonomous Underwater Vehicles are considered as hybrid vehicles between ROVs and AUVs but not completely. In fact, IAUVs can carry out manipulation task of ROVs integrated in AUVs (without tether) but the poor qualities of acoustic communication link complexifies the teleoperation process and reduces its performance.
5. **Glider Vehicles (GVs)**: An underwater glider uses buoyancy variation and 'yoyo' depth control as propulsion mean. Attitude is controlled by adaptive redistribution of mass or external control surfaces. The advantages of GVs are long range missions, low-cost operations.
6. **Bio-inspired Vehicles (Bio-Vs)**: The fact that living species show optimal performance through natural selection, therefore, numerous Bio-Vs have been developed in recent years to follow biological shapes. Inheriting flexibility of natural systems, most of Bio-Vs can maneuver by changing their shapes.

Generally, control of underwater system raises questions in 6 Degree of Freedoms (DoFs) (roll, pitch, yaw, surge, sway, and heave), but most of applications just only require the control of 3+1 DoFs (surge, heave, and pitch, yaw), depending on the actuation capacity of the system and objectives of missions.

From actuation point of view, an underwater vehicle can be classified into an under-actuated, iso-actuated, or over-actuated system. This depends on the number and position of actuators carried by the vehicle and the DoFs that this actuation system can impact. Note that a system can be under-actuated even if it carries more actuators than DoFs. This will be detailed in Chapter 4. Among them, an over-actuated system has some advantages such as explicit management of the redundancy for robustness, compensation of actuation defaults (dead-zone) [Ropars 2015], and hybrid systems (AUV/ROV). In this thesis, over-actuated properties of an AUV are exploited for karst exploration which is a confined environment.

1.3 The karst stakes

As aforementioned, karst generally comprises a network of underground natural conduits, resulting from the dissolution of soluble rocks, limestone, dolomite and gypsum (Figure 1.2). These aquifers drain groundwater on a large scale, from their inland catchment basin to their marine exsurgences. They supply drinking water to millions of people worldwide and, during heavy rainfall, may host violent transfers of charge that can cause dramatic and sudden floods in fragile and unpredictable areas.

The urgent need for management tools of underground resources requires a precise knowledge of the underlying conduit network, in terms of position, depth, geomorphology, and seasonal and episodic dynamics. Exploitation of this resource requires precise drilling which must penetrate these conduits in a region with an appropriate morphology (pumping chamber), in order to reply to the pumping

demand, also considering the seasonal variability of the resource availability and quality.

Hydrogeological risk management requires having a precise knowledge of the hydrosystem dynamics, running models in order to predict floods occurrence, or afford this underground network with a dam flood control role [Jourde 2007]

It is thus of major importance to get reliable information about the position, geometry and dynamics of these karstic networks along their entire development, from inland to marine resurgence. This is a crucial and urgent issue for public authorities in charge of prospection, protection, and active management of the groundwater resource in karstic regions.

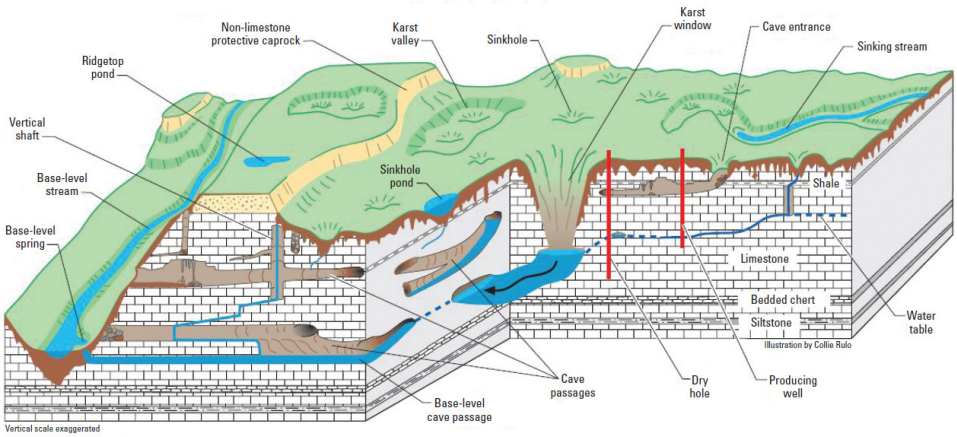


Figure 1.2 – Karst Topology [Taylor 2008]

1.4 A rapid history of karst exploration with robots

The first document reporting on the use of an underwater robot in karst environment is about the Télénaute, in 1967, operated by the IFP (Institut Français du Pétrole) who pushed the exploration of Fontaine de Vaucluse down to 106m. This robot was fully teleoperated and initiated the edifying story of the robotic exploration of this legendary spring, as Roland Pastor reports [FontaineDeVaucluse] (Figure 1.3), and from which the following description is largely inspired. In 1983, a new generation of systems, Sorgonaute, was built in collaboration with Renault. It took over Télénaute, and reached 243m depth, but was stopped by the length of its cable. The year after, Sorgonaute II was lost at 233m because its thrusters were trapped in a remaining lifeline. In 1986, Sorgonaute III was built to recover its predecessor. The mission turned nightmare and the system was lost with 150m of cumbersome floating cable. Sorgonaute IV, in 1988, tried a rescue mission that also failed. Finally, the systems were recovered, and the site cleared, by divers. This raises the question of the appropriateness of such risk taking to recover defective robots. Besides being a brilliant failure, this adventure is, above all, a highly valu-

able experimental feedback. In the meantime, Modexa, from the company MIC, reached the deepest point at 315m and discovered the entrance of a large horizontal karst development.

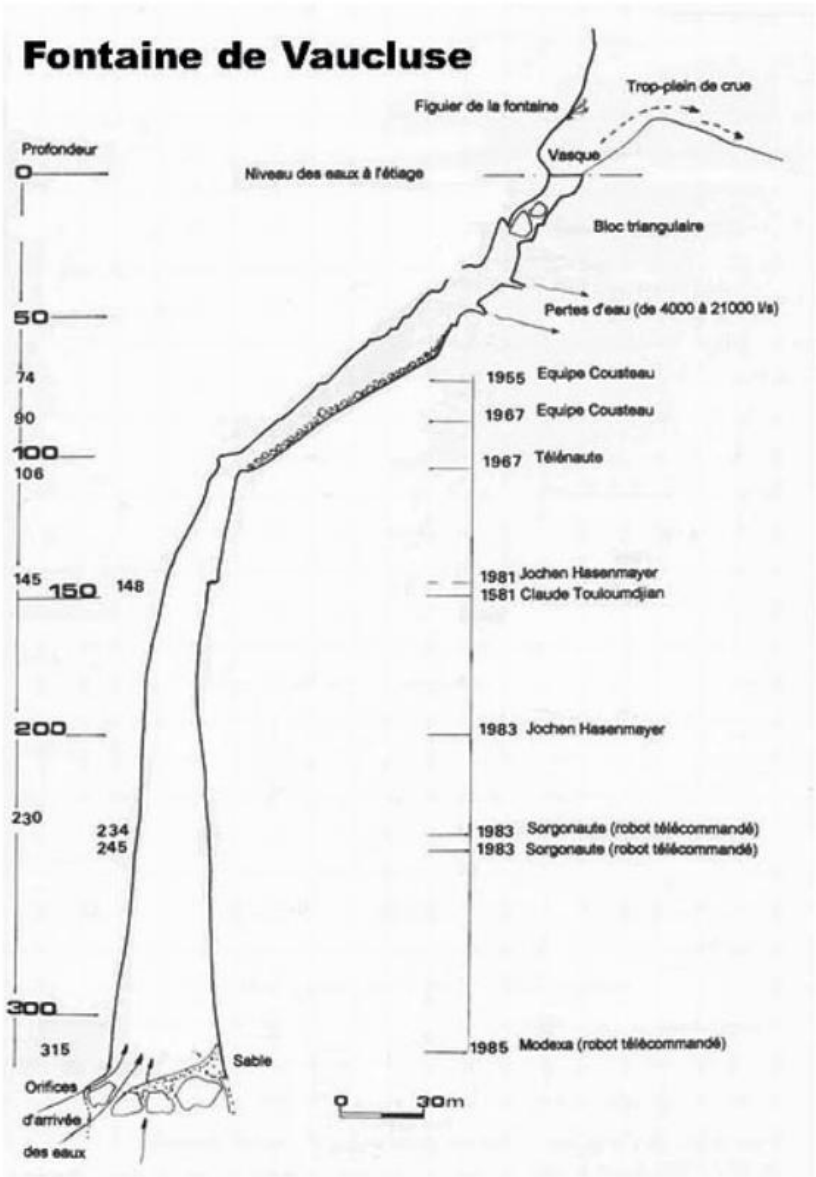


Figure 1.3 – Fontaine de Vaucluse Topology [FontaineDeVaucluse]

These systems were essentially huge teleoperated by video systems, without a specific sensor suite. But they solved a lot of technological issues (depth, mobility),

Figure 1.4 – Télénaute (left) Sorgonaute (center) Spélénaute (right) [Lapierre 2016]

which now benefit to current systems. In 1989, the Spélénaute was built by COMEX [Comex], with a higher ambition, benefiting from recent advances in acoustic sensor technology. It was designed with a complete navigation sensor-suite (compass, depth-gauge, echosounder, and current meter) to build the first realistic topography of the chasm, and reached the bottom as Modexa did, 4 years before. Télénaute, Sorgonaute, and Spélénaute robots are shown in Figure 1.4.

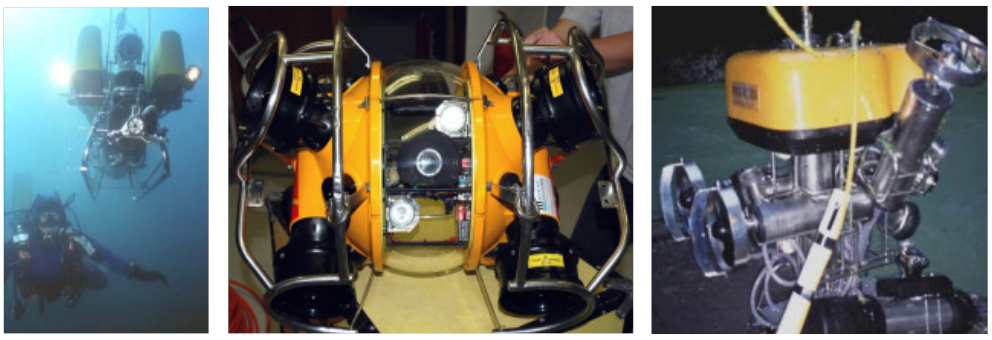


Figure 1.5 – Mercury (left) Hyball (center) Prometheus (right) [Lapierre 2016]

The previous story is centered on Fontaine de Vaucluse, a commune in France, but similar experiment was made all over Europe. As illustrative example, we can highlight the exploration of the deepest vertical cave in the world [1], the chasm "Pozzo del Merro", where the Mercury robot, operated by the Italian MSTD team [2], made the first exploration in 2000, reaching 210m depth. Few months later, the Hyball reached 310m, approaching the world record held by Modexa, but without touching down. Finally, in 2002, the Prometheus, owned by the Milanese Firemen Corps [3], established the actual record at 392m depth, within this emblematic chasm, and discovered a narrow horizontal continuation (Figure 1.5).

Previous systems are ROV (Remotely Operated Vehicle) which load a necessary,

1. https://web.infinito.it/utenti/s/simonant/merro-english.htm
2. Mediterranean Sea Technical Divers: https://web.infinito.it/utenti/s/simonant/index.htm
3. http://www.vigilfuoco.it/sitiVVF/milano/notizia.aspx?codnews=14235s=281

but highly problematic, umbilical cable. Autonomy is quite difficult to achieve in such condition. Nevertheless, some AUV (Autonomous Underwater Vehicle) were developed. Among them, the most advanced system is the DEPTHX (DEep Phreatic THermal eXplorer) developed, in 2005, by University American Consortium and sponsored by NASA, and dedicated to Cenote [4] exploration [Stone 2007]. DEPTHX is approximately 1.5m tall and 1.9m in length and width for a 1.5 ton weight. The vehicle has a full suite of underwater navigation sensors and a network of 46 discrete sonar elements. It was designed to navigate in unexplored environments, generate high resolution 3-D maps, collect biological samples, and return autonomously to its origin. Besides the system integration challenge that represents the conception of such a complete system, the development focuses on the on-line 3-D mapping and the SLAM navigation question [Fairfield 2006], which provided in 2007 a complete cartography [5] of the Cenote La Pilita [Fairfield 2007], northern Mexico. Worthwhile noticing, an experimental validation of an autonomous sample collection was performed, and opened the quite challenging extraterrestrial microbial life investigation on the Jovian moon Europa [Sahl 2010] (Figure 1.6).

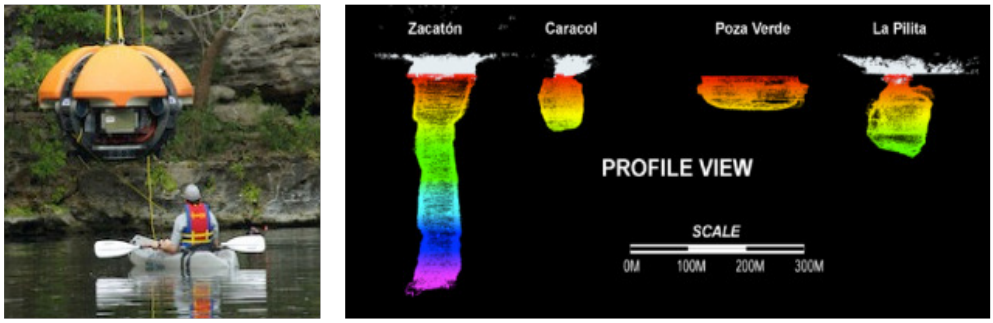


Figure 1.6 – The DEPTHX (left) and the 3D mapping of different cenotes (right)

The DEPTHX system is the result of a very nice integration challenge, but is almost dedicated to vertical exploration. Moreover, its heavy logistics disqualifies this system for routinely operations that the karst exploration challenge requires. Previous work was oriented to patrimonial, depth competition, and of course science, but where not facing the current and vital societal challenge of understanding water resource dynamics.

The Unexmin European Project [6] proposes to design and develop autonomous systems for exploration and mapping of Europe's flooded mines, in order to provide authorities pertinent information to re-active the exploitation of these abandoned mines. The focus is on the question of miniaturization and adaptation of deep-sea robotics technology to this new application environment and to the interpretation of geoscientific data [Lopes 2017]. the main distinguishing feature of this flooded

4. Cenote is a sinkhole formed by volcanic activity eroding the carbonate rock.
5. 3D reconstruction model: https://www.facebook.com/stoneaerospace/videos/1245970192284/
6. https://www.unexmin.eu/

mine environment, with respect to karst, is the man-made, hence structured conditions of the environment (in some cases known a priori based on historical maps) and the fact that the spatial dimensions of many galleries allow for some free room to maneuver without jeopardizing the integrity of the robot. Given the confined environment of the karst and the quest for autonomous exploration of uncharted underground networks, the results of Unexmin and their contribution to help solving some of the karst-related exploration problems are eagerly expected. Other systems dedicated to 3D mapping of underwater caves have been developed, as man-driven systems, where mapping sensors are mounted on an underwater scooter, such as the Digital Wall Mapper System developed by B. Stone [Ende 2001], which was used to prefigure the sensors suite of the DEPTHX and the Sunfish HROV systems. The latter is announced to be a commercial autonomous underwater cave mapper, but only some information has been published about the Sunfish HROV [Richmond 2018].

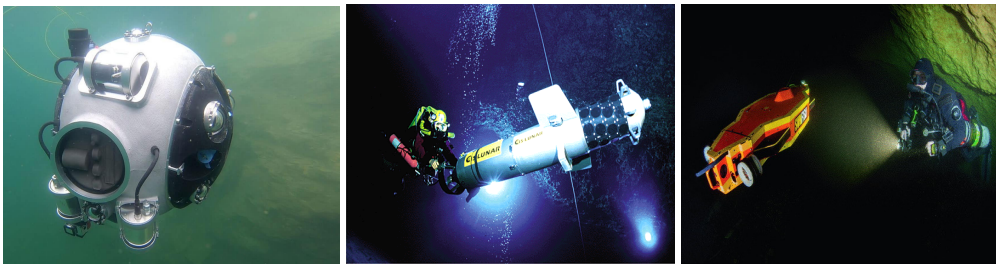


Figure 1.7 – Unexmin (left) Digital wall mapper (center) Sunfish (right)

Finally, the ROV Ulysse and the manned system Navscoot were developed in Aleyin Project, a local funded project from the Labex Numev of University of Montpellier. Ulysse is equipped with a profiling sonar, a complete suite of navigation sensors (IMU and DVL) and carries 12 thrusters. Navscoot is a man-driven system obtained by mounting the sensor suite of Ulysse on an underwater scooter. With these systems, an acoustic 3D topography and a partial photogrammetric reconstruction have been made [Lasbouygues 2017] (Figure 1.8).

Routine in karst exploration conceals many opened technological and scientific issues, which are exposed in the sequel.

1.5 Concepts for karst exploration and Robotic challenges

Karst exploration that we have chosen, in which the present study is included, is depicted in Figure 1.9 and divided into two phases: *exploration phase and return phase* (from ALEYIN project [7] which supports this research activity, including the funding of this PhD program).

7. ALEYIN project was supported by LabEx NUMEV (ANR-10-LABX-0020) within the I-SITE MUSE (ANR-16-IDEX-0006) and the Region Occitanie (French FEDER funds)

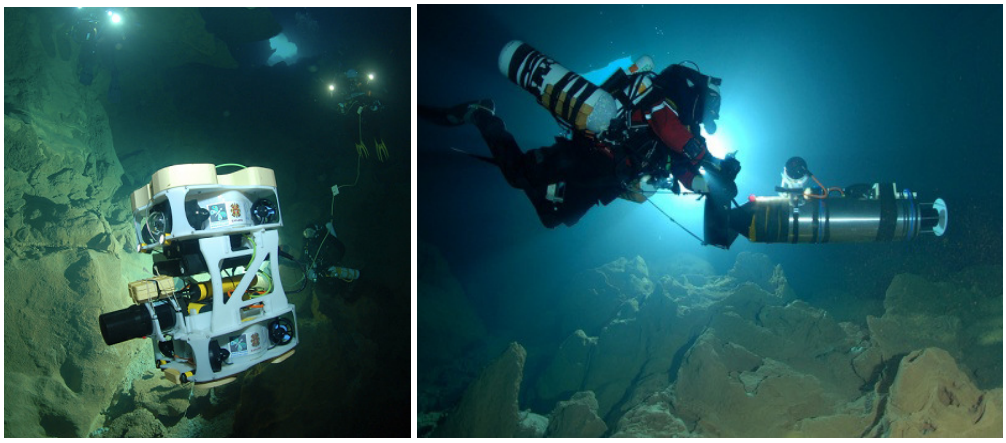


Figure 1.8 – Ulysse (left) Underwater scooter (right)

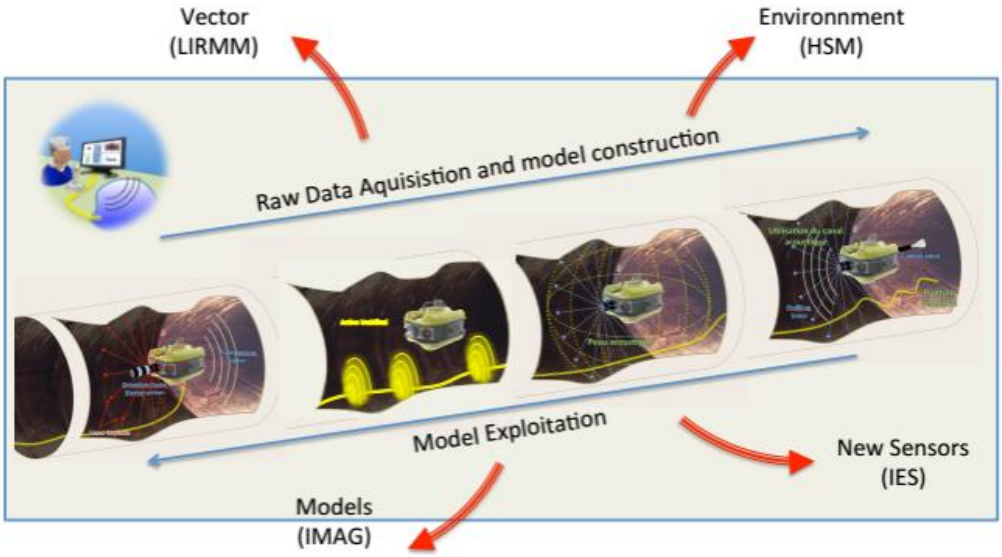


Figure 1.9 – Karst exploration concepts (ALEYIN project): French institutes: LIRMM, HSM, IMAG, IES

1. **Exploration phase**: Because of complexity of karst terrain and low bandwidth of underwater acoustic sensors, the presence of the umbilical cable is mandatory and an expert needs to be in the loop. In this phase, robot operates in a ROV configuration. However, it is not realistic to imagine ROV being able to drag an umbilical cable along kilometres of chaotic relief. It drives a interesting question of umbilical cable management which is discussed in the sequel. In control perspective, in a confined context, the full teleoperation of a ROV is a difficult task. Hence, a potential solution

relies on co-control strategy, where the system autonomously performs the control of given degrees of freedom, while the remaining ones stay under the operator control. An effective solution, during exploration phase, is that the robot is autonomously centered, using acoustic or video sensors, insuring its own safety, while the operator controls its progression and attitude. Note that this requires reactive control architecture in which a suite of sensors or techniques can collect environmental data with good enough of sampling period. Following Figure 1.9, the objectives of this phase are to collect raw data and build a model which is useful for the return phase. To this end, the system is driven closer to the environment, in order to precisely observe a region of interest. In this context, observation sensors can be advantageously used for safe observation.

2. **Return phase**: In this phase, the robot release its umbilical cable and switches to AUV configuration. It will also exploit data collected during exploration phase. The objective is safely homing. In particular, robot will use raw data of the exploration phase to extract useful information to build 3D map in order to home with Simultaneous Localization and Mapping (SLAM) techniques.

Following these two phases in karst exploration, challenging problems and interesting ideas exploiting a robot safety and effectively are revealed:

1. **Umbilical cable management**: An onboard motorized secable truncanner is necessary, but it is also complex and delicate mechatronic device which requires a particular attention. Moreover, the presence of this umbilical cable, as reported by previous karst exploration attempts, is the major cause of failure since the cable is highly subject to things being blocked within the relief of the environment, specially during the return phase. In this phase, it has to be noticed that the cable, even unplugged, is equipping to the environment, and could be advantageously used as the diver's lifeline. This implies to develop an active cable that guides the homing navigation.
2. **Navigation and Mapping**: Navigation and Mapping are quite important in karst environment where GPS does not work and navigation techniques for open water (LBL, USBL, GIB, and single beacon navigation) can not be used. SLAM techniques is suitable to this issue. Nevertheless, efficient SLAM algorithm with respect to on-board computational limitation of robot for kast exploration remains an open question.
3. **Guidance and Control**: Leveraging on techniques that have been developed for ROVs and AUVs, a co-control architecture should be required. However, an effective and complete integration system is always a challenging problem.
4. **Robustness**: This is one of pivotal requirements for a confined environment exploration. For minimizing performance degradation and avoiding dangerous operational conditions in the case of actuator failures, fault detection

and tolerance strategies can be applied. Beyond all, it is redundancy of robot which can be exploited to compensate actuator limitations or avoid failures in physical level.

5. **Reactivity and adaptable autonomy**: Facing with unpredictable issues (varying current, salinity variation), and the complexity of conduit section in karst network, reactivity management and adaptable autonomy of robot are extremely important. Moreover, different modes of observation (local observation, long course) are carried out during a mission. A varying geometry of actuation system will be a potential solution. Furthermore, energy consumption can be saved with an adaptive configuration.

As aforementioned discussions, many scientific challenges for robotics are derived in karst exploration with respect to performance and safety of missions. In fact, confronting with the complexity of conduit section (chaotic topology, closed space, decreasing and increasing of section) which can yield unforeseeable issues such as varying frontal current and causes difficulties in robot missions. Another problem is that the robot has to save its energy as much as possible because of battery's limitation. This requires a robust, reactive, and adaptable design of robot's configuration and redundant system is a good choice. ***This thesis focuses on architecture in propulsion perspective and configuration design of redundant AUVs which optimize objective functions in order to deal with these requirements***.

1.6 Autonomous Underwater Robot Architecture and Configuration

This section discusses briefly about architectures of AUVs in propulsion perspective, with static or dynamic configurations. A detailed survey of them is presented in Chapter 4 and Chapter 6.

1.6.1 Static configuration

Most of current UVs have a static configuration of their actuators (can not change during missions). Specifically, shape and propulsion system of UVs are fixed. Static configuration UVs have advantages and drawbacks also. For example, torpedo-shaped AUVs can be powerful for long range and high speed missions, however they are not suitable for operations in narrow spaces, low speeds, and station-keeping. Dynamic configuration UVs can overcome limitations by switching the configuration of their actuation system. However, a flexible system implies other challenging questions in control and control allocation, for instance. This will be discussed in the sequel. Although Bio-Vs have fixed configurations but we can classify into dynamic configurations because they are able to modify their shapes during missions. For static configurations, a challenging questions is how to optimize the design with respect to some 'well-chosen' performance criteria. The

thesis deals with this problem in some perspectives and more details are presented in Chapter 4.

1.6.2 Dynamic configuration

UVs with dynamic configurations have many advantages in comparison with ones with static configurations. Indeed, dynamic configuration UVs are able to adapt their configuration according to mission requirements. Bio-Vs belong to this category, as explained in the previous section. However, it is not easy to model and control such UVs because of their hyper-redundant nature which requires specific mathematical tools and control strategies [Lapierre 2006b]. There exits numerous questions for a dynamic configuration system, for example, when and how to change configuration in order to guarantee control performances, smooth transition between configurations, optimal configuration for specified mission, etc. This thesis also addresses this issue in Chapter 6 and Chapter 7.

1.7 Motivations and Contributions of the thesis

This thesis focuses on redundant systems in underwater robot field applied to karst and marine exploration. This work focuses on configuration design and related problems: control allocation and control design. In particular, the positions and orientations of the thrusters are firstly determinated with static configuration in which performance indices are proposed and multiobjective optimization approach is used to find a solution. This yields an interesting question when positions and orientations of thrusters can be changed dynamically. Therefore, a dynamic configuration robot is studied by building a prototype and investigating its properties.

The contributions of the thesis are summarized as follows:

- *Propose the criteria, manipulability, energetic, workspace, reactive, and robustness indices, to design an optimal static configuration for over-actuated underwater robots. Multiobjective optimization methods are investigated and goal attainment approach is chosen to solve the problem. Simulation and experiments results show the efficiency of the proposed approach.*
- *Build a prototype of AUVs with reconfigurable configuration, called Umbrella Robot. The design procedure from hardware to software and Inertia Measurement Unit (IMU) calibration are presented. Simulations and experiments are carried out on the robot. And then, an optimal configuration problem w.r.t geometry distances is proposed to improve the robot's performances.*
- *Propose optimal dynamic configuration problem for Umbrella Robot. This method exploits the advantage of dynamic configuration of the robot with respect to energy-like criterion. In the meantime, control allocation methods are studied for this case. Simulation results are shown to prove the feasibility of the method.*

1.8 Structure of the thesis

The thesis is composed with nine chapters. The main points of each chapter are shortly presented as follows:

1. **Chapter 1**: Chapter 1 presents important roles of karst system and ocean in human life and the needs of their exploration. The overview of underwater robot domain is also investigated with classification of robots and a rapid history of using robot for karst exploration is presented. The ideas and challenging questions for karst exploration are discussed. Finally, motivation and contributions of the thesis are also established.
2. **Chapter 2**: Fundamentals in underwater robotics, from kinematic to dynamic models, quaternion representation, nonlinear control strategies, and control allocation are briefly presented in this part.
3. **Chapter 3**: Optimization and multiobjective optimization techniques are shown in this chapter. These are basic tools for solving proposed problems in the following chapters.
4. **Chapter 4**: This chapter presents static configuration design problem for over-actuated underwater robots. Performance indices are proposed and analyzed. A multiobjecitve optimization method is chosen and procedure of searching for optimal solution is proposed.
5. **Chapter 5**: Simulations and experimental results of static configuration design problem are presented. A comparison between two configurations are shown to prove the proposed approach.
6. **Chapter 6**: A prototype of an underwater robot with reconfigurable configuration, called Umbrella Robot, is built. The reconfigurable capability is analyzed. Acting abilities along/about 6 DoFs are proposed and optimization problem with respect to geometry distances are suggested.
7. **Chapter 7**: A problem of robot's dynamic configuration is presented in this chapter. An algorithm, called A-SQP which is based on Sequential Quadratic Programming, is proposed to solve the problem.
8. **Chapter 8**: Simulation and experiment results for Umbrella Robot were shown in the chapter. For dynamic configuration, simulation results with different issues, i.e., given missions, path following, and observation case, are studied.
9. **Chapter 9**: Conclusions and future works are drawn.

Chapter 2

Fundamentals in Underwater Robotics

Contents

Fundamentals for underwater robotics, from kinematic to dynamic models, quaternion representation, nonlinear control strategies, and control allocation review are briefly presented in this part.

2.1 Underwater Robot model

This section presents background of underwater vehicles, from common notations to kinematic and dynamics models.

2.1.1 Notation

All common and primary notations in marine vehicle's motion are given in this part as in Figure 2.1. Specific ones will be shown when needed. Let's define the 2 reference frames:

1. $\{I\}$ denotes the NED (North-East-Down) inertial frame
2. $\{B\}$ denotes a direct frame attached to the vehicle's metacenter (Body-frame)

According to these frames, the following notations will be used:

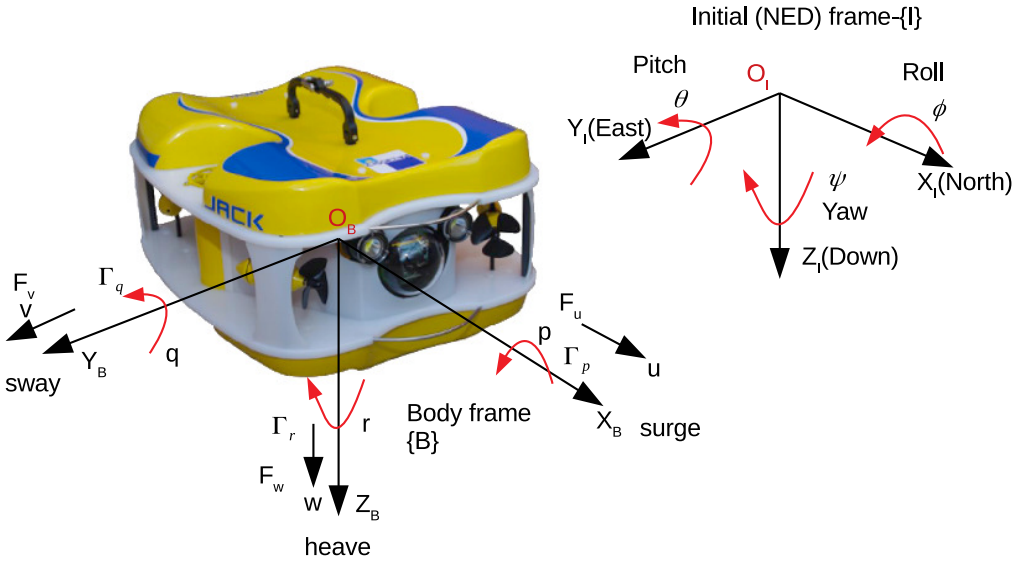


Figure 2.1 – Notations of marine vehicle's motion

1. $\boldsymbol{\eta} = [\boldsymbol{\eta_1} \quad \boldsymbol{\eta_2}]^T$ denotes the system states within the inertial frame $\{I\}$, where:
   — $\boldsymbol{\eta_1} = [x \quad y \quad z]^T$ denotes the system's position in the inertial frame $\{I\}$.
   — $\boldsymbol{\eta_2} = [\phi \quad \theta \quad \psi]^T$ denotes the system's attitude (orientations) w.r.t $\{I\}$, using Euler angles.
2. $\mathbf{Q}$ denotes the system's attitude w.r.t $\{I\}$, using quaternion formalism.
3. $\boldsymbol{\nu} = [\boldsymbol{\nu_1} \quad \boldsymbol{\nu_2}]^T$ denotes the system's velocities expressed in $\{B\}$, where:
   — $\boldsymbol{\nu_1} = [u \quad v \quad w]^T$ denotes the system's linear velocities w.r.t $\{I\}$ and expressed in $\{B\}$.
   — $\boldsymbol{\nu_2} = [p \quad q \quad r]^T$ denotes the system's angular velocities around each axis of $\{B\}$.
4. $\mathbf{W} = [0, \boldsymbol{\nu_2}^T]$ denotes the system's angular velocities expressed in $\{B\}$ with quaternion formalism.
5. $\mathbf{F}_B = [\mathbf{F} \quad \boldsymbol{\Gamma}]^T$ denotes the resulting actions of the actuation system, expressed in $\{B\}$, where:
   — $\mathbf{F} = [F_u \quad F_v \quad F_w]^T$ denotes the resulting forces of the actuation system expressed in $\{B\}$.
   — $\boldsymbol{\Gamma} = [\Gamma_p \quad \Gamma_q \quad \Gamma_r]^T$ denotes the resulting torques of the actuation system expressed in $\{B\}$

2.1.2 Kinematic model

The kinematic model expresses the kinematic relation between inertial and body frames, that the system constrains (none in our case since our system is holonomic).

The expression of this model depends, of course, of the chosen formalism. Euler formalism is known for embedding the 'Gimbal Lock' singularity, which occurs for $\theta \approx \pm\frac{\pi}{2}$. Considering that most of the applications can be performed with a small pitch (for aerial and marine drones) the Euler formalism is largely used. Its principal interest is to manipulate comprehensive elements: roll, pitch and yaw, as a heritage from marine science. However, the quaternion formalism relaxes the Gimbal lock singularity. Both approaches are presented in the following.

2.1.2.1 Euler formalism

For easily following, the kinematic model is divided into two parts: translation and rotation. First, the relation of time derivative of the position vector $\boldsymbol{\eta_1}$ and linear velocity vector $\boldsymbol{\nu_1}$ via the following transformation:

$$\dot{\boldsymbol{\eta}}_1 = \mathbf{J}_1(\boldsymbol{\eta}_2)\boldsymbol{\nu}_1 \tag{2.1}$$

where $\mathbf{J}_1(\boldsymbol{\eta}_2)$ is a rotation matrix relating to the functions of Euler angles. This matrix is given by:

$$\mathbf{J}_1(\boldsymbol{\eta_2}) = \begin{pmatrix} c(\psi)c(\theta) & -s(\psi)c(\phi) + s(\phi)s(\theta)c(\psi) & s(\psi)s(\phi) + s(\theta)c(\psi)c(\phi) \\ s(\psi)c(\theta) & c(\psi)c(\phi) + s(\phi)s(\theta)s(\psi) & -c(\psi)s(\phi) + s(\theta)s(\psi)c(\phi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{pmatrix} \tag{2.2}$$

where $c() = cos()$ and $s() = sin()$.

On the other hand, the relation of time derivative of the Euler angles vector $\boldsymbol{\eta_2}$ and angular velocity vector $\boldsymbol{\nu_2}$ via the transformation:

$$\dot{\boldsymbol{\eta}}_2 = \mathbf{J}_2(\boldsymbol{\eta}_2)\boldsymbol{\nu}_2 \tag{2.3}$$

where $\mathbf{J}_2(\boldsymbol{\eta}_2)$ is a transformation matrix (Jacobian matrix) is given by:

$$\mathbf{J}_2(\boldsymbol{\eta_2}) = \begin{pmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & \frac{s(\phi)}{c(\theta)} & \frac{c(\phi)}{c(\theta)} \end{pmatrix} \tag{2.4}$$

where $c() = cos()$, $s() = sin()$, and $t() = tan()$.

Combining (2.1) and (2.3) yields the kinematics mode of ocean vehicles in compact form:

$$\dot{\boldsymbol{\eta}} = \mathbf{J}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{2.5}$$

where $\boldsymbol{\eta} = [\boldsymbol{\eta_1} \quad \boldsymbol{\eta_2}]^T = [x \quad y \quad z \quad \phi \quad \theta \quad \psi]^T$ and $\boldsymbol{\nu} = [\boldsymbol{\nu_1} \quad \boldsymbol{\nu_2}]^T = [u \quad v \quad w \quad p \quad q \quad r]^T$. The full transformation matrix is as:

$$\mathbf{J}(\boldsymbol{\eta}) = \begin{pmatrix} \mathbf{J}_1(\boldsymbol{\eta_2}) & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{J}_2(\boldsymbol{\eta_2}) \end{pmatrix} \tag{2.6}$$

For avoiding singularities or Gimbal locks phenomena in kinematic model, quaternions representation is alternative approach which is presented in the sequel.

2.1.2.2 Quaternion formalism

Basic definitions[Kuipers 1999]: A quaternion $\mathbf{Q}$ is defined as hyper-complex numbers with one real part and three imaginary parts as:

$$\mathbf{Q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} \tag{2.7}$$

where $q_0, q_1, q_2, q_3 \in \mathbb{R}$, and $\mathbf{i}, \mathbf{j}, \mathbf{k}$ denote the orthonormal basic for $\mathbb{R}^3$.

An any unit quaternion can be described as:

$$\mathbf{Q} = [\cos\frac{\alpha}{2}, \sin\frac{\alpha}{2}.\mathbf{n}^T]^T \tag{2.8}$$

where $\mathbf{n}$ is a unit vector, and α is an angle (we can see more details of these two parameters in quaternion rotation).

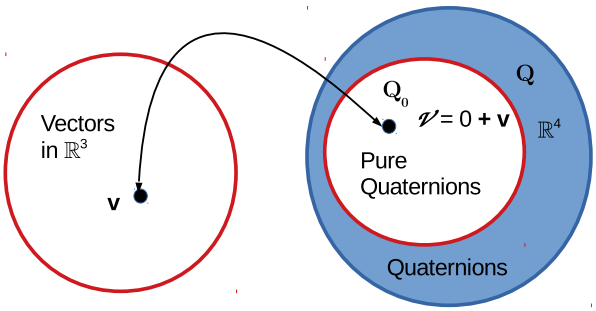


Figure 2.2 – mapping: vector and quaternion spaces

Basic operations (add, multiply, conjugate, norm, inverse, and derivative) can be referred to Appendix A.2.

Quaternion and Rotation

Theorem 2.1.1 *[Kuipers 1999] For any unit quaternion, we can describe:* $\mathbf{Q} = (\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\mathbf{n})$ *and for any vector* $\mathbf{V} \in \mathbb{R}^3$*, the action of the operator:*

$$L_{\mathbf{Q}}(\mathbf{v}) = \mathbf{Q} \odot \mathbf{V} \odot \mathbf{Q}^*$$

on $\mathbf{V}$ *is equivalent to a rotation of the vector through an angle* θ *about* $\mathbf{n}$ *as the axis of rotation.*

Theorem 2.1.2 *[Kuipers 1999] For any unit quaternion, we can describe:* $\mathbf{Q} = (\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\mathbf{n})$ *and for any vector* $\mathbf{V} \in \mathbb{R}^3$*, the action of the operator:*

$$L_{\mathbf{Q}}(\mathbf{v}) = \mathbf{Q}^* \odot \mathbf{V} \odot \mathbf{Q}$$

on $\mathbf{V}$ *is equivalent to*

- *a rotation of the coordinate frame about the axis* $\mathbf{n}$ *through an angle* θ *while* $\mathbf{V}$ *is not rotated, or,*

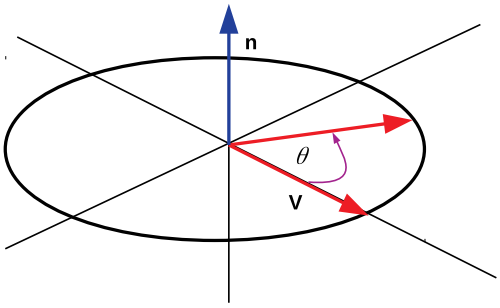


Figure 2.3 – Rotation with quaternion

- *an opposite rotation of the vector* $\mathbf{V}$ *w.r.t the coordinate frame through an angle* θ *about the axis* $\mathbf{n}$.

Note that Theorem 2.1.1 and Theorem 2.1.2 can be considered as the same belonging the choice of rotation angle θ. The analysis details were presented in [Kuipers 1999].

Kinematics

Let $\mathbf{Q}$ be a unit quaternion, and $\boldsymbol{\omega}$ is angular velocity of the rotated frame ($\boldsymbol{\omega}$ is the angular velocity expressed in body frame), and $\mathbf{W} = [0 \quad \boldsymbol{\omega}]^T$, the derivative of $\mathbf{Q}$ is:

$$\dot{\mathbf{Q}} = \frac{1}{2}\mathbf{Q} \odot \mathbf{W} \tag{2.9}$$

Equation 2.9 is considered as rotational kinematic model with quaternion formalism. For translational kinematic model, Theorem 2.1.1 is useful, it is given by:

$$\dot{\boldsymbol{\eta}}_{1Q} = \begin{bmatrix} 0 \\ \dot{\boldsymbol{\eta}}_1 \end{bmatrix} = \mathbf{Q} \odot \begin{bmatrix} 0 \\ \boldsymbol{\nu}_1 \end{bmatrix} \odot \mathbf{Q}^* \tag{2.10}$$

Full kinematic model of a marine vehicle with Euler and quaternion formalism is summarized in Table 2.1.

Unwinding phenomenon

In particular, reaching a desired quaternion, robot can rotate through clockwise direction or vice versa. This yields undesirable behavior, called unwinding phenomenon, in designing a continuous closed-loop feedback control that stabilize a rotational motion (topological obstruction) [Bhat 2000] [Chaturvedi 2011].

Finally, transformations between rotation matrix or Euler angles and quaternions can be referred to Appendix A.2.

<table>
<tr><td rowspan="3">Pose</td><td>Translation</td><td colspan="2">$\boldsymbol{\eta_1} = [x \quad y \quad z]^T$</td><td>$\boldsymbol{\eta_{1Q}} = [0 \quad x \quad y \quad z]^T$</td></tr>
<tr><td rowspan="2">Rotation</td><td colspan="2">Euler</td><td>Quaternion</td></tr>
<tr><td colspan="2">$\boldsymbol{\eta_2} = [\phi \quad \theta \quad \psi]^T$</td><td>$\mathbf{Q}$</td></tr>
<tr><td rowspan="3">Velocities</td><td>Translation</td><td colspan="3">$\boldsymbol{\nu_1} = [u \quad v \quad w]^T$</td></tr>
<tr><td rowspan="2">Rotation</td><td colspan="2">Euler</td><td>Quaternion</td></tr>
<tr><td colspan="2">$\boldsymbol{\nu_2} = [p \quad q \quad r]^T$</td><td>$\mathbf{W} = [0 \quad \boldsymbol{\nu_2}^T]^T$</td></tr>
<tr><td rowspan="2">Kinematic model</td><td>Euler</td><td colspan="3">Quaternion</td></tr>
<tr><td>$\dot{\boldsymbol{\eta}} = \begin{bmatrix} \dot{\boldsymbol{\eta}}_1 \\ \dot{\boldsymbol{\eta}}_2 \end{bmatrix} = \mathbf{J}(\boldsymbol{\eta_2}) \begin{bmatrix} \boldsymbol{\nu_1} \\ \boldsymbol{\nu_2} \end{bmatrix}$</td><td colspan="3">$\begin{bmatrix} \dot{\boldsymbol{\eta}}_{1Q} \\ \dot{\mathbf{Q}} \end{bmatrix} = \begin{bmatrix} \mathbf{Q} \odot \begin{bmatrix} 0 \\ \boldsymbol{\nu_1} \end{bmatrix} \odot \mathbf{Q}^* \\ \frac{1}{2}\mathbf{Q} \odot \mathbf{W} \end{bmatrix}$</td></tr>
</table>

Table 2.1 – Pose, Velocity representations, and Kinematic model with Euler and Quaternion

2.1.3 Dynamic model

Dynamics model presents the relation between derivative of linear and angular velocities and external forces and torques along/about each axes. For simplicity, we do not consider ocean current velocity in the following equations. Dynamic model of a marine vehicle is given by [Fossen 2011]:

$$\mathbf{M}\dot{\boldsymbol{\nu}} = \mathbf{F}_B + \mathbf{F}_{wind} + \mathbf{F}_{wave} - \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} - \mathbf{g}(\boldsymbol{\eta}) \tag{2.11}$$

where $\mathbf{M} = \mathbf{M}_{RB} + \mathbf{M}_A$, $\mathbf{M}$ is rigid-body mass matrix, $\mathbf{M}_A$ is hydrodynamic added mass matrix; $\mathbf{C} = \mathbf{C}_{RB} + \mathbf{C}_A$, $\mathbf{C}_{RB}$ is rigid-body Corriolis-Centripetal matrix due to the rotation of body-frame $\{B\}$ about inertial frame $\{I\}$, $\mathbf{C}_A$ is Corriolis-Centripetal matrix of added mass; $\mathbf{D} = \mathbf{D}_l + \mathbf{D}_n$ with $\mathbf{D}_l$ and $\mathbf{D}_n$ are linear and nonlinear damping matrix respectively, $\mathbf{F}_{wind}$ and $\mathbf{F}_{wave}$ are environmental forces/moments (wind and wave). $\mathbf{F}_B$ is applying forces/moments w.r.t body-frame. Finally, $\mathbf{g}(\boldsymbol{\eta})$ is buoyancy forces and torques. For more details, we can refer to Appendix B.

2.2 Nonlinear control methods

As it has been clearly stated in [Brockett 1983] any system which exhibits a non-holonomic constraint cannot be stabilized with linear controller. This is the case for the well known 'unicycle' which is used as a generic system to address the question of movement regulation. Considering marine systems, as shown in [Lapierre 2007], there are many advantages to extend the control solutions for unicycle-type systems to underactuated (marine) systems. One should note that an iso-actuated system can be stabilize using linear approaches. However, our concern here focuses on system with varying capacities of the actuation system. This implies that, according to its actuation configuration, the system can evolve from iso-actuation to under-actuation (and vice versa), visiting during its continuous evolution all the intermediate states, potentially ill-conditioned. Hence, the appropriate control de-

sign is nonlinear. We present in the following some nonlinear approaches that can be adopted to control such a system.

This section presents briefly nonlinear control methods for nonlinear systems. In general, a nonlinear system is given as:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \tag{2.12a}$$

$$\mathbf{y} = h(\mathbf{x}) \tag{2.12b}$$

where $\mathbf{x} \in \mathbb{R}^n$ is a state vector of the system, $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{y} \in \mathbb{R}^p$ are control input and output vectors respectively. f and h are nonlinear functions.

The objective is to design control input, $\mathbf{u}$, to satisfy control performances. In the following sections, some typical nonlinear control methods are illustrated for the system (2.12). The author does not have the ambition to survey all nonlinear control methods for underwater robots and only to present briefly nonlinear control methods for nonlinear systems.

2.2.1 Gain-scheduling method

Gain-scheduling method is an extension of the linearization approach, which is often used at one operating point, to a range of operating points. The concept of gain scheduling stems from flight control system. The method is suitable for systems which can parameterize the operating points by one or more variables, called scheduling variables. In this case, a system will be linearized at several equilibrium points and each linear feedback controller at each point is designed. A family of linear controllers, in compact form as gain scheduled controller, is implemented with corresponding scheduling variables. The procedure of designing a gain scheduled controller as follows [Khalil 2002]:

1. Linearize the nonlinear model about a family of operating points, parameterized by the scheduling variables.
2. Design each linear controller to achieve the specified performance at each operating point.
3. Construct a gain-scheduled controller such that:
   (a) for each constant value of the exogenous input.
   (b) the linearization of the closed-loop system under the gain-scheduled controller is equivalent to the linearization of the closed-loop system under the fixed-gain controller.
4. Check the nonlocal performance of the nonlinear closed-loop model.

It is obvious that gain-scheduled method inherits methods from linear systems.

2.2.2 Feedback linerization

The basic idea of feedback linearization is to cancel nonlinearities by linearization which employs a change of coordinates and feedback control to tranform a

nonlinear system to a system whose dynamics are linear. Feedback linearization is different from the conventional linearization which is a linear approximation of dynamics. Indeed, an example for input-output linearization case, considering a class of nonlinear system (2.12) is given by:

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u} \tag{2.13a}$$

$$\mathbf{y} = h(\mathbf{x}) \tag{2.13b}$$

The derivative of the output $\mathbf{y}$ is given by:

$$\dot{\mathbf{y}} = \frac{\partial h}{\partial \mathbf{x}}(\mathbf{x})f(\mathbf{x}) + \frac{\partial h}{\partial \mathbf{x}}(\mathbf{x})g(\mathbf{x})\mathbf{u} \tag{2.14}$$

Assuming that $\frac{\partial h}{\partial \mathbf{x}}(\mathbf{x})g(\mathbf{x}) = \mathbf{0}$, we define:

$$\psi_1(\mathbf{x}) = h(\mathbf{x}), \quad \psi_2 = \frac{\partial h}{\partial \mathbf{x}}(\mathbf{x})f(\mathbf{x}) \tag{2.15}$$

We have the second derivative of $\mathbf{y}$:

$$\ddot{\mathbf{y}} = \frac{\partial}{\partial \mathbf{x}}(\frac{\partial h}{\partial \mathbf{x}}f)f + \frac{\partial}{\partial \mathbf{x}}(\frac{\partial h}{\partial \mathbf{x}}f)g\mathbf{u} = \frac{\partial \psi_2}{\partial \mathbf{x}}(\mathbf{x})f(\mathbf{x}) + \frac{\partial \psi_2}{\partial \mathbf{x}}(\mathbf{x})g(\mathbf{x})\mathbf{u} \tag{2.16}$$

Similarly, assuming that $\frac{\partial \psi_2}{\partial \mathbf{x}}(\mathbf{x})g(\mathbf{x}) = \mathbf{0}$, we continue the differential procedure to p times:

$$\mathbf{y}^p = \frac{\partial \psi_p}{\partial \mathbf{x}}(\mathbf{x})f(\mathbf{x}) + \frac{\partial \psi_p}{\partial \mathbf{x}}(\mathbf{x})g(\mathbf{x})\mathbf{u} \tag{2.17}$$

If $\frac{\partial \psi_p}{\partial \mathbf{x}}(\mathbf{x})g(\mathbf{x})\mathbf{u} \neq \mathbf{0}$ (this relates to relative degree definition of system (2.13) [Krstic 1995]) and $\mathbf{u}$ is chosen as:

$$\mathbf{u} = \frac{1}{\frac{\partial \psi_p}{\partial \mathbf{x}}(\mathbf{x})g(\mathbf{x})}[-\frac{\partial \psi_p}{\partial \mathbf{x}}(\mathbf{x})f(\mathbf{x}) + \mathbf{v}] \tag{2.18}$$

We have:

$$\mathbf{y}^p = \mathbf{v} \tag{2.19}$$

Then the dynamics of $\mathbf{y}$ and its derivatives are governed by a chain of p integrators. It is clear that the system is input-output linearizable because of the state feedback control (2.18) which reduces input-output map to (2.19) and new input v is designed with strategies of linear systems.

Another case is state feedback linearization, readers can refer to [Khalil 2002] and [Isidori 1989]. With feedback linearization technique, we can inherit control design methods from linear systems.

2.2.3 Backstepping

The backstepping method is a recursive design procedure. This method is based on Lyapunov stability theorem, which derives *control Lyapunov function*. Consider a nonlinear state space system as Equation (2.12a) and a virtual control $\mathbf{v}$ is given by:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{v}) \tag{2.20a}$$

$$\dot{\mathbf{v}} = g(\mathbf{v}, \mathbf{u}) \tag{2.20b}$$

The backstepping method is designed recursively. Virtual control $\mathbf{v}$, Equation (2.20a), is designed to satisfy control performances, for example in kinematic stage. Afterwards, control input $\mathbf{u}$ is designed with virtual control $\mathbf{v}$ from previous step. The order of designing complexity depending on kind of feedback systems: strict-feedback systems, pure-feedback systems, and block-strict-feedback systems, is increased. General design procedures can be referred in [Krstic 1995], and applications in marine systems can be found in [Lapierre 2008].

2.2.4 Model predictive control

Model Predictive Control (MPC) is a control method which directly considers input constraints into design procedure. It stems from optimal control problem, which optimizes an objective function. The main challenge of MPC is to solve online optimization problem. Because MPC methods is based on numerical computations, it is more convenient to express Equation (2.12a) in discrete form as:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \tag{2.21a}$$

For defining the MPC scheme, a stage cost function, $l(\mathbf{x}_k, \mathbf{u}_k)$, is defined. It can be a penalizing function of the distance from equivalent point and control effort, e.g., $l(\mathbf{x}_k, \mathbf{u}_k) = \|\mathbf{x}_k - \mathbf{x}^*\| + \lambda\|\mathbf{u}_k\|$. The MPC problem defines an optimization problem which minimizes objective function, $V(\mathbf{x}, \mathbf{u}, N)$ as:

$$\min_{\mathbf{u}} V(\mathbf{x}, \mathbf{u}, N) = \sum_{k=1}^{N-1} l(\mathbf{x}_k, \mathbf{u}_k) + V(\mathbf{x}_N) \tag{2.22a}$$

$$s.t \quad \mathbf{x}_k = f(\mathbf{x}_k, \mathbf{u}_k) \tag{2.22b}$$

$$\mathbf{x}_0 = \mathbf{x0} \tag{2.22c}$$

$$\mathbf{u} \in \mathbb{U} \tag{2.22d}$$

where N is the control horizon, $\mathbf{x0}$ is the initial value, and $\mathbb{U}$ is the input constraints.

MPC approach has been studies in recent years, especially in 2000's and has many applications in industry especially in chemical industry where response time of system is not too fast. Readers can refer to [Rawlings 2017] and references therein for more details. A fast and efficient tool for implementing MPC methods is CasADi [Andersson 2019].

2.3 Error function for control with Quaternion

Designing controller with quaternion formalism can avoid Gimbal lock and it is chosen to build controllers of our robots in this thesis. This section shows a primary theorem in quaternion error which is deployed in quaternion-based controller.

Theorem 2.3.1 *Quaternion error between current quaternion* $\mathbf{Q}$ *and desired quaternion* $\mathbf{Q}_d$ *is defined as:*

$$\mathbf{Q}_e = \mathbf{Q}_d^* \odot \mathbf{Q}$$

then

$$\dot{\mathbf{Q}}_e = \frac{1}{2}(-\mathbf{W}_d \odot \mathbf{Q}_e + \mathbf{Q}_e \odot \mathbf{W})$$

where $\mathbf{W}_d = 2.\mathbf{Q}_d^* \odot \dot{\mathbf{Q}}_d$ *is desired angular velocity (in quaternion form), and* $\mathbf{W} = [0 \quad \boldsymbol{\omega}]^T$.

Proof First, note that all quaterions are unit. Following Theorem 5.3 in [Kuipers 1999] for two consecutive rotations, we have:

$$\mathbf{Q} = \mathbf{Q}_d \odot \mathbf{Q}_e \tag{2.23}$$

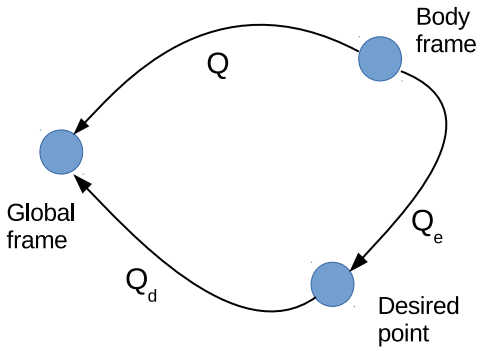


Figure 2.4 – Consecutive rotations with quaternion

Multiplying both sides by inverse of $\mathbf{Q}_d$

$$\mathbf{Q}_e = \mathbf{Q}_d^{-1} \odot \mathbf{Q} \tag{2.24}$$

or

$$\mathbf{Q}_e = \mathbf{Q}_d^* \odot \mathbf{Q} \tag{2.25}$$

We prove the derivative of quaternion error.

$$\dot{\mathbf{Q}}_e = \dot{\mathbf{Q}}_d^* \odot \mathbf{Q} + \mathbf{Q}_d^* \odot \dot{\mathbf{Q}} \tag{2.26}$$

$$= \dot{\mathbf{Q}}_d^* \odot \mathbf{Q}_d \odot \mathbf{Q}_e + \frac{1}{2}\mathbf{Q}_d^* \odot \mathbf{Q} \odot \mathbf{W} \tag{2.27}$$

$$= \frac{1}{2}(\mathbf{W}_d^* \odot \mathbf{Q}_e + \mathbf{Q}_e \odot \mathbf{W}) \tag{2.28}$$

$$= \frac{1}{2}(-\mathbf{W}_d \odot \mathbf{Q}_e + \mathbf{Q}_e \odot \mathbf{W}) \tag{2.29}$$

where $\mathbf{W}_d = 2.\mathbf{Q}_d^* \odot \dot{\mathbf{Q}}_d$, and note that $\mathbf{W}_d^* = -\mathbf{W}_d$

This completes the proof. ∎

2.4 Control Allocation

In control engineering, a system is designed in order to accomplish one specific task or more. In general, it is considered as under-actuated system, iso-actuated system, or over-actuated one. This depends to the numbers of actuators, their geometric structure, and the degree of freedoms (DOFs) of system. The over-actuated systems are redundant systems (note that vice-versa direction is not completely right as discussion in Chapter 4, indeed, some over-actuated systems can have under-actuated behavior.). Redundant systems have been researched and developed for many years because of their advantages, especially in uncertain and disturbance environments.

The basic property of a redundant system is that the number of necessary actuators is higher than DOFs. The problem is how to map desired actuation on DOFs to forces on actuators. In literature, there are two approaches to solve this problem. The first method is to divide the control design to high level and low level. In the high level, we consider that a system is fully actuated, this means that we can design control laws for each DOF. The outputs of high level block, called virtual controls, are the inputs of low level block. In the low level, a control allocation algorithm is designed to assign control inputs for actuators in order to optimize one or some cost functions. This problem is called control allocation problem. The second method is that the control inputs (normally with constraints) are directly taken into account in control design process. It is obvious to see this issue in MPC method because control allocation is considered as constraints in MPC formulation. However, this yields to computational cost which is the most challenging task in MPC problem.

Control allocation problem is one of the main tasks in control design of redundant systems. Normally, actuators of system are constrained with mechanical and electrical limitations, thus saturations have to be taken into account in control design. The role of control allocation in control loop is displayed in Figure 2.5.

There are a lot of control allocation methods, from linear to nonlinear, from unconstrained to constrained. Without constraints, the problem is easier. However, unconstrained control allocation problem is the basic ideas for many constrained control allocation problem. In recent years, probabilistic techniques were also used

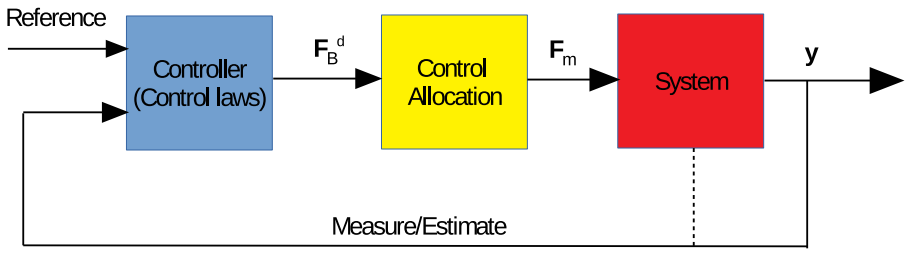


Figure 2.5 – A general control loop

in control allocation problem [Elliott 2016]. Most of the control allocation methods are based on optimization techniques, explicitly or implicitly. Therefore, computational cost is also the challenging task in control allocation problem for complex systems. Depending on applications, suitable control allocation method is chosen. Sometimes, we have to compromise between the computational cost and performances.

There are some survey works on control allocation problem in recent years. In [Page 2000], the authors compare many control allocation algorithms with closed-loop and open-loop measures. In [Bodson 2002], the author evaluates the performance and computational cost of optimization methods of control allocation problem. In [Fossen 2006] and [Fossen 2009], control allocation methods for ships and underwater vehicles were investigated. A survey paper was published in 2013 [Johansen 2013], in which many control allocation methods and applications were presented and discussed.

In marine vehicle domain, the principle of control allocation methods (in Figure 2.5) is written as:

$$\mathbf{F}_B = \mathbf{A}.\mathbf{F}_m \tag{2.30}$$

where $\mathbf{F}_B \in \mathbb{R}^n$ is acting vector along/about DoFs in body-frame, $\mathbf{A} \in \mathbb{R}^{n \times m}$ ($n = 6$ DoFs) is configuration matrix, and $\mathbf{F}_m \in \mathbb{R}^m$ is force vector on actuators. Note that, in this case, propulsion system is expressed as a configuration matrix.

The objective is, with a given configuration matrix $\mathbf{A}$, desired actuation demand $\mathbf{F}_B^d$ (output from a controller), and Equation (2.30), how to find the control forces to be applied by actuators $\mathbf{F}_m$ which satisfy constraints of actuators (saturation and dead-zone) and make $\mathbf{F}_B$ close to $\mathbf{F}_B^d$ as much as possible.

Normally, there are two classes of Control Allocation (CA) methods. The first one is based on the Moore-Penrose pseudo inverse including Direct method, Daisy-chain method [Durham 1993], Cascade General Inverse (CGI) [Bordignon 1996], and the second one is based on optimization techniques such as sequential least square, minimal least square, and fixed-point method, nonlinear programming method [Härkegård 2003]. One class of CA methods based on neural network [Skulstad 2018] has been proposed recently. Many studies have been published to solve the control

allocation problem. Readers can refer to [Johansen 2013] and references therein for more details.

We consider the first class of CA methods. From (2.30), we derive:

$$\mathbf{F}_m = \mathbf{A}^+ \mathbf{F}_B^d \tag{2.31}$$

where $\mathbf{A}^+$ denotes as Moore-Penrose pseudo inverse of $\mathbf{A}$ matrix.

In order to consider the saturation of actuators, Direct method, Daisy-chain and CGI method were proposed. A variety of (2.31) proposed in [Ropars 2015] to avoid dead-zones of thruster's characteristics (thruster is used as actuator in marine vehicle) is given as:

$$\mathbf{F}_m = \mathbf{A}^+ \mathbf{F}_B^d + \mathbf{r}_m \tag{2.32}$$

where $\mathbf{r}_m \in Ker(\mathbf{A})$ is called the *common regime*.

Basically, the problem of CA methods based on nonlinear programming is formulated as:

$$\min_{\mathbf{F}_m} \|\mathbf{F}_B^d - \mathbf{A}\mathbf{F}_m\| \tag{2.33a}$$

$$s.t \quad \mathbf{F}_m \in \mathbb{F} \tag{2.33b}$$

where $\mathbb{F}$ is the constraint set of actuator forces.

A taxonomy of CA methods is shown in Figure 2.6.

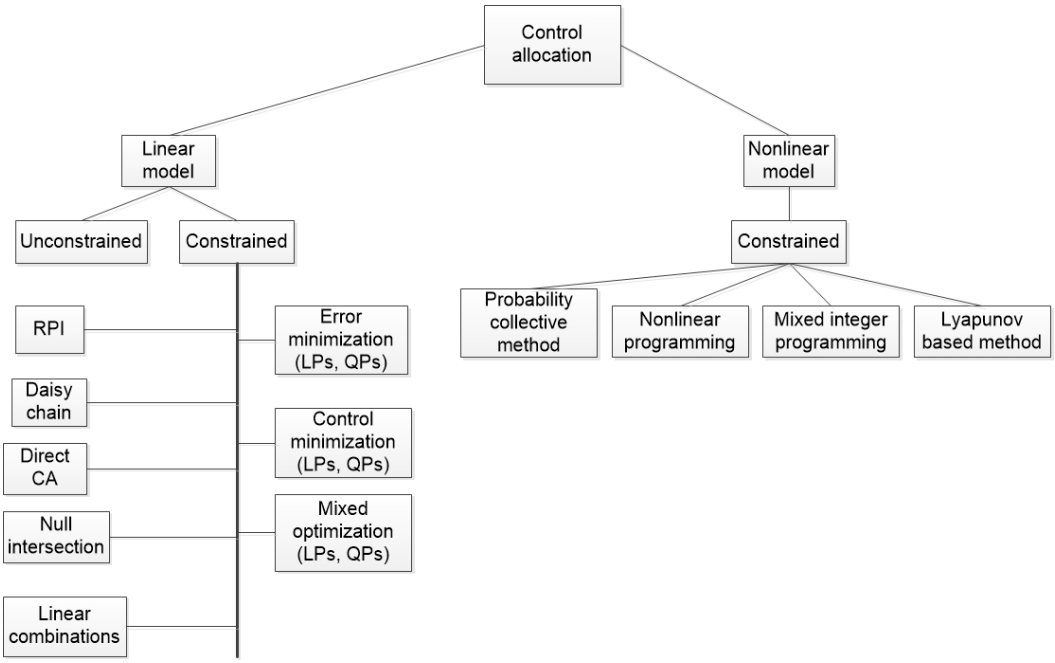


Figure 2.6 – Taxonomy of CA methods

2.5 Conclusion

This chapter briefly presented fundamentals to study underwater vehicles, especially for redundant autonomous underwater robots. First, kinematic and dynamic models in 6 DoFs of underwater vehicles were shown in Euler angles formalism and quaternion formalism which is mainly used in the thesis. Secondly, nonlinear control methods, including gain-scheduled, feedback linearization, backstepping, and model predictive control (MPC), which spread out in wide range of techniques (from inheriting control methods for linear systems (gain-scheduled) to applying optimization in control theory (MPC)) were summarized. Among them, backstepping method was used in designing controller for our robots. Deploying advantages of designing a controller with quaternion formalism (avoid gimbal lock, appear in compact form - $SO(3)$ space), a theorem in quaternion error, which is a guidance of controller design in the thesis, was presented and proved. Furthermore, control allocation, a mapping from desired control vector ($\mathbf{F}_B^d$) to desired thruster force vector ($\mathbf{F}_m^d$), plays important role in studies of redundant systems and then a review of control allocation approaches was shown in the last of the chapter. These fundamentals are applied in the next chapters such as kinematic and dynamic models and control allocation methods for simulations in Chapter 7, Quaternion control for real robot in Chapter 8.

Chapter 3

Optimization and Multiobjective Optimization

Contents

This chapter shows fundamentals in optimization and multiobjective optimization techniques. Specifically, Sequential Quadratic Programming (SQP) with active-set method is presented. A review of multiobjective optimization methods are given out. These are basic tools for solving proposed problems in the following chapters.

3.1 Optimization

This section introduces backgrounds of constrained nonlinear optimization problem which are based on [Nocedal 2006] [Boyd 2004]. This is also basic theory in solving multiobjective optimization problem which is used in Chapter 4. Moreover, Sequential Quadratic Programming is the baseline of proposed algorithm in Chapter 7. Note that in this chapter, optimization problem is minimization problem and maximization one can be transformed to minimization problem.

3.1.1 Problem definition

The general constrained optimization problem is defined as:

$$\min_{\mathbf{x}} f(\mathbf{x}) \tag{3.1a}$$

$$s.t \quad c_i(\mathbf{x}) = 0, i \in \mathcal{E} \tag{3.1b}$$

$$c_i(\mathbf{x}) \leq 0, i \in \mathcal{I} \tag{3.1c}$$

where $\mathbf{x} \in \mathbb{R}^n$ is vector variable, $f(\mathbf{x}) \in \mathbb{R}$ is scalar objective function, $c_i(\mathbf{x})$ is i^{th} constraint, $\mathcal{E}$ and $\mathcal{I}$ are index spaces of equality and inequality respectively.

3.1.2 Basic definitions and theories

Definition 1 *An inequality constraint $c_i(\mathbf{x}) \leq 0$ is said to be **active** at $\mathbf{x}^*$ if $c_i(\mathbf{x}) = 0$. Otherwise, it is said to be **inactive** if $c_i(\mathbf{x}) < 0$.*

Definition 2 *The active set $\mathcal{A}(\mathbf{x})$ at any feasible $\mathbf{x}$ consists of the equality constraints indices from $\mathcal{E}$ together with the indices of inequality constraints i for which $c_i(\mathbf{x}) = 0$, that is $\mathcal{A}(\mathbf{x}) = \mathcal{E} \cup \{i \in \mathcal{I} | c_i(\mathbf{x}) = 0\}$.*

Definition 3 *A point $\mathbf{x}$ is called a **regular point** if the set of active constraint gradients $\nabla c_i(\mathbf{x}), i \in \mathcal{A}(\mathbf{x})$ is linearly independent.*

Theorem 3.1.1 *(First-order necessary conditions-KKT (Karush-Kuhn-Tucker) conditions) [Nocedal 2006] Let the functions f and c_i be continuously differentiable and $\mathbf{x}^*$ is a regular point and a local minimizer for the problem (3.1). Then, there exists a vector of Lagrangian multipliers $\boldsymbol{\lambda}^*$ with elements λ_i^* such that:*

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) = \mathbf{0} \tag{3.2a}$$

$$c_i(\mathbf{x}^*) = 0, i \in \mathcal{E} \tag{3.2b}$$

$$c_i(\mathbf{x}^*) \leq 0, i \in \mathcal{I} \tag{3.2c}$$

$$\lambda_i^* \geq 0, i \in \mathcal{I} \tag{3.2d}$$

$$\lambda_i^* c_i(\mathbf{x}^*) = 0, i \in \mathcal{I} \tag{3.2e}$$

Theorem 3.1.2 *(Second-order necessary conditions) [Nocedal 2006] Suppose that $\mathbf{x}^*$ is a local solution and a regular point of problem (3.1). Let $\boldsymbol{\lambda}^*$ be the Lagrangian multipliers vector for which KKT conditions are satisfied. Then:*

$$\mathbf{w}^T \nabla^2_{\mathbf{xx}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) \mathbf{w} \geq 0 \quad \textit{for all} \quad \mathbf{w} \in \mathcal{C}(\mathbf{x}^*, \boldsymbol{\lambda}^*) \tag{3.3}$$

3.2 Optimization methods

There exits numerous approaches to solve nonlinear constraint optimization problem [Nocedal 2006] [Boyd 2004] [Bertsekas 1995]. This section focuses on active-set method of Sequential Quadratic Programming (SQP), which is efficient for small and medium problem, because it is used to solve optimization formulation of dynamic configuration robot in Chapter 7.

3.2.1 Quadratic programming with equality constraints

This section presents how to solve quadratic programming problem with equality constraints. This is the basic in active-set method for SQP. The Quadratic Programming (QP) problem is defined as:

$$\min_{\mathbf{x}} f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{x}^T\mathbf{c} \tag{3.4a}$$

$$s.t \quad \mathbf{A}\mathbf{x} = \mathbf{b} \tag{3.4b}$$

where $\mathbf{x} \in \mathbb{R}^n$ is variable vector, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{H} \in \mathbb{R}^{n\times n}$.

The Lagrangian function of this problem is $\mathcal{L} = f(\mathbf{x}) + \boldsymbol{\lambda}^T(\mathbf{A}\mathbf{x} - \mathbf{b})$ with $\boldsymbol{\lambda} \in \mathbb{R}^m$ is vector of Lagrangian multipliers. The KKT conditions is written as:

$$\nabla_{\mathbf{x}}\mathcal{L} = \mathbf{0} \tag{3.5}$$

$$\mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0} \tag{3.6}$$

The KKT conditions can be detailed as:

$$\mathbf{H}\mathbf{x}^* + \mathbf{c} + \mathbf{A}^T\boldsymbol{\lambda}^* = \mathbf{0} \tag{3.7}$$

$$\mathbf{A}\mathbf{x}^* - \mathbf{b} = \mathbf{0} \tag{3.8}$$

This can be rewritten in matrix form:

$$\begin{pmatrix} \mathbf{H} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{x}^* \\ \boldsymbol{\lambda}^* \end{pmatrix} = \begin{pmatrix} -\mathbf{c} \\ \mathbf{b} \end{pmatrix} \tag{3.9}$$

Equation (3.9) can be solved by techniques of nonlinear equations: direct or iterative methods. For more details, readers can refer into [Nocedal 2006].

If we replace $\mathbf{x}^* = \mathbf{x}_k + \mathbf{p}$ ($\mathbf{x}_k$ is a value at iteration k and $\mathbf{p}$ is a direction) into Equation (3.9), we have:

$$\begin{pmatrix} \mathbf{H} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{p} \\ \boldsymbol{\lambda}^* \end{pmatrix} = \begin{pmatrix} -\mathbf{c} - (\mathbf{H}\mathbf{x}_k + \mathbf{A}^T\mathbf{x}_k) \\ \mathbf{b} - \mathbf{A}\mathbf{x}_k \end{pmatrix} = \begin{pmatrix} \mathbf{g} \\ \mathbf{h} \end{pmatrix} \tag{3.10}$$

With Equation (3.10), we can find the optimal direction ($\mathbf{p}$) for the problem at an iteration. Matrix factorization methods are used to solve this Equation.

3.2.2 Quadratic programming with active set method

This section extends the previous part for QP problem in general case, i.e. with equality and inequality constraints. The problem is written as:

$$\min_{\mathbf{x}} f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{x}^T\mathbf{c} \tag{3.11a}$$

$$s.t \quad \mathbf{a}_i^T\mathbf{x} = b_i, \quad i \in \mathcal{E} \tag{3.11b}$$

$$\mathbf{a}_i^T\mathbf{x} \leq b_i, \quad i \in \mathcal{I} \tag{3.11c}$$

where $\mathbf{a}_i \in \mathbb{R}^n, b_i \in \mathbb{R}$. $\mathcal{E}$ and $\mathcal{I}$ are index sets of equality and inequality constraints respectively.

The Lagrangian function of this problem $\mathcal{L} = f(\mathbf{x}) + \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i (\mathbf{a}_i^T \mathbf{x} - b_i)$

KKT conditions for this problem are:

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) = \mathbf{0} \tag{3.12a}$$
$$\mathbf{a}_i^T \mathbf{x}^* - b_i = 0, i \in \mathcal{E} \tag{3.12b}$$
$$\mathbf{a}_i^T \mathbf{x}^* - b_i \leq 0, i \in \mathcal{I} \tag{3.12c}$$
$$\lambda_i \geq 0, i \in \mathcal{I} \tag{3.12d}$$
$$\lambda_i (\mathbf{a}_i^T \mathbf{x}^* - b_i) = 0, i \in \mathcal{I} \tag{3.12e}$$

Specializing the KKT conditions with the definition of active set $\mathcal{A}(\mathbf{x}^*)$, the KKT conditions can be reformulated as:

$$\mathbf{H}\mathbf{x}^* + \mathbf{c} + \sum_{i \in \mathcal{A}(\mathbf{x}^*)} \lambda_i \mathbf{a}_i = \mathbf{0} \tag{3.13a}$$
$$\mathbf{a}_i^T \mathbf{x}^* - b_i = 0, i \in \mathcal{A}(\mathbf{x}^*) \tag{3.13b}$$
$$\mathbf{a}_i^T \mathbf{x}^* - b_i < 0, i \in \mathcal{I} \setminus \mathcal{A}(\mathbf{x}^*) \tag{3.13c}$$
$$\lambda_i \geq 0, i \in \mathcal{I} \cap \mathcal{A}(\mathbf{x}^*) \tag{3.13d}$$
$$\lambda_i = 0, i \in \mathcal{I} \setminus \mathcal{A}(\mathbf{x}^*) \tag{3.13e}$$

Suppose that at k^{th} iteration, the active set at $\mathbf{x}_k$ is known in advance, called $\mathcal{W}_k$. How can we solve the quadratic programming problem (3.11).

To find the direction of line search at k^{th} iteration, $\mathbf{p}$, we replace $\mathbf{x} = \mathbf{x}_k + \mathbf{p}$ into (3.11a):

$$f(\mathbf{x}) = f(\mathbf{x}_k + \mathbf{p}) = \frac{1}{2}(\mathbf{x}_k + \mathbf{p})^T \mathbf{H}(\mathbf{x}_k + \mathbf{p}) + (\mathbf{x}_k + \mathbf{p})^T \mathbf{c} \tag{3.14a}$$
$$= \frac{1}{2}\mathbf{p}^T \mathbf{H}\mathbf{p} + \mathbf{p}^T(\mathbf{H}\mathbf{x}_k + \mathbf{c}) + \frac{1}{2}\mathbf{x}_k^T \mathbf{H}\mathbf{x}_k + \mathbf{x}_k^T \mathbf{c} \tag{3.14b}$$
$$= \frac{1}{2}\mathbf{p}^T \mathbf{H}\mathbf{p} + \mathbf{p}^T \mathbf{h}_k + \rho_k \tag{3.14c}$$

where $\mathbf{h}_k = \mathbf{H}\mathbf{x}_k + \mathbf{c}$ and $\rho_k = \frac{1}{2}\mathbf{x}_k^T \mathbf{H}\mathbf{x}_k + \mathbf{x}_k^T \mathbf{c}$ (independent of $\mathbf{p}$, so we can drop this term)

We define a QP sub-problem:

$$\min_{\mathbf{p}} \frac{1}{2}\mathbf{p}^T \mathbf{H}\mathbf{p} + \mathbf{p}^T \mathbf{h}_k \tag{3.15a}$$
$$s.t \quad \mathbf{a}_i^T \mathbf{p} = 0, i \in \mathcal{W}_k \tag{3.15b}$$

We can use any techniques in QP problem with equality constraints to solve the problem (3.15). Assume that the solution of (3.15) is $\mathbf{p}_k$, we have $\mathbf{a}_i^T \mathbf{x} = \mathbf{a}_i^T(\mathbf{x}_k + \alpha_k \mathbf{p}_k) = \mathbf{a}_i^T \mathbf{x}_k = b_i$ for all $i \in \mathcal{W}_k$. So the constraint (3.13b) which satisfied at $\mathbf{x}_k$ is also satisfied when we move along direction $\mathbf{p}_k$.

We consider if the search direction $\mathbf{p}_k$ is nonzero, we have to decide how far to move along this direction. If $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k$ is feasible with all constraints, we set $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k$. Otherwise, we set:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \tag{3.16}$$

where α_k is the step-length parameter is chosen to be the largest value in the range $[0, 1]$ for which all constraints are satisfied.

Considering constraint (3.12c)(constraints with $i \notin \mathcal{W}_k$), we have:

$$\mathbf{a}_i^T \mathbf{x}_{k+1} = \mathbf{a}_i^T (\mathbf{x}_k + \alpha_k \mathbf{p}_k) = \mathbf{a}_i^T \mathbf{x}_k + \alpha_k \mathbf{a}_i^T \mathbf{p}_k \tag{3.17}$$

If $\mathbf{a}_i^T \mathbf{p}_k \leq 0$, the constraint (3.12c) is satisfied for all $\alpha_k \in [0, 1]$ because $\mathbf{a}_i^T \mathbf{x}_k < b_i$. If $\mathbf{a}_i^T \mathbf{p}_k > 0$, we choose $0 < \alpha_k < \frac{b_i - \mathbf{a}_i^T \mathbf{x}_k}{\mathbf{a}_i^T \mathbf{p}_k}$, the constraint (3.12c) will be also satisfied.

To maximize the decrease of objective function, we want α_k to be as large as possible in $[0, 1]$, so we have a formula to choose step-length parameter as follows:

$$\alpha_k = \min \left(1, \min_{i \notin \mathcal{W}_k, \mathbf{a}_i^T \mathbf{p}_k > 0} \frac{b_i - \mathbf{a}_i^T \mathbf{x}_k}{\mathbf{a}_i^T \mathbf{p}_k} \right) \tag{3.18}$$

We continue to iterate in this manner, adding constraints to the working set until we reach an optimal point $\hat{\mathbf{x}}$ which minimizes the quadratic objective function over its current working set $\hat{\mathcal{W}}$. But how to recognize that point?. In fact, at that point, the QP sub-problem (3.15) has the solution $\mathbf{p} = 0$. Applying KKT conditions for the problem (3.15) with $\mathbf{p} = 0$, we have:

$$\begin{aligned} \sum_{i \in \hat{\mathcal{W}}} \hat{\lambda}_i \mathbf{a}_i + \mathbf{h}_k &= 0 \\ \sum_{i \in \hat{\mathcal{W}}} \hat{\lambda}_i \mathbf{a}_i + \mathbf{H}\mathbf{x}_k + \mathbf{c} &= 0 \end{aligned} \tag{3.19}$$

for some Lagrangian multipliers $\hat{\lambda}_i, i \in \hat{\mathcal{W}}$. In order to satisfy the constraint (3.13d), we need $\hat{\lambda}_i \geq 0$ with $i \in \mathcal{I} \cap \hat{\mathcal{W}}$.

The algorithm for QP problem with active set method is presented in **Algorithm** 1

3.3 Sequential Quadratic Programming

Now, we consider the nonlinear general problem (3.1). We linearize objective function, equality and inequality constraints to model:

$$\min_{\mathbf{p}} f_k + \nabla_{\mathbf{x}} f_k^T \mathbf{p} + \frac{1}{2} \mathbf{p}^T \nabla_{\mathbf{xx}}^2 \mathcal{L}_k \mathbf{p} \tag{3.20a}$$

$$s.t \quad \nabla_{\mathbf{x}} c_i(\mathbf{x}_k)^T \mathbf{p} + c_i(\mathbf{x}_k) = 0, \quad i \in \mathcal{E} \tag{3.20b}$$

$$\nabla_{\mathbf{x}} c_i(\mathbf{x}_k)^T \mathbf{p} + c_i(\mathbf{x}_k) \leq 0, \quad i \in \mathcal{I} \tag{3.20c}$$

Algorithm 1 Active set algorithm for QP problem [Nocedal 2006]

Initialize a feasible starting point $\mathbf{x}_0$, maxIter
Set $\mathcal{W}_0$ to be a active set at $\mathbf{x}_0$
for $k = 0, 1, 2... \leq maxIter$ **do**
    Solve (3.15) using Algorithms for QP problem with equality constraints to find $\mathbf{p}_k$
    if $\mathbf{p}_k = \mathbf{0}$ **then**
        Compute corresponding Lagrangian multipliers, $\hat{\lambda}_i$, which satisfy (3.19) with $\hat{\mathcal{W}} = \mathcal{W}_k$
        if $\hat{\lambda}_i \geq 0$ for all $i \in \mathcal{I} \cap \mathcal{W}_k$ **then**
            Stop with solution $\mathbf{x}^* = \mathbf{x}_k$
        else
            $j = \arg\min_{j \in \mathcal{I} \cap \mathcal{W}_k} \hat{\lambda}_j$
            $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k$
            $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{j\}$
        end if
    else
        Compute step-length α_k from (3.18)
        $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{p}_k$
        if there are blocking constraints ($\alpha_k < 1$) **then**
            Obtain $\mathcal{W}_{k+1}$ by adding one of blocking constraints to $\mathcal{W}_k$
        else
            $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k$
        end if
    end if
end for

3.3.1 Quasi-Newton approximations

To approximate the Hessian matrix, $\nabla^2_{\mathbf{xx}}\mathcal{L}_k$, we define:

$$\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k \tag{3.21}$$

$$\mathbf{y}_k = \nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}_{k+1}, \boldsymbol{\lambda}_{k+1}) - \nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}_k, \boldsymbol{\lambda}_{k+1}) \tag{3.22}$$

With given $\mathbf{B}_k$, we define:

$$\theta_k = \begin{cases} 1 & \text{if } \mathbf{s}_k^T\mathbf{y}_k \geq 0.2\mathbf{s}_k^T\mathbf{B}_k\mathbf{s}_k \\ \dfrac{0.8\mathbf{s}_k^T\mathbf{B}_k\mathbf{s}_k}{\mathbf{s}_k^T\mathbf{B}_k\mathbf{s}_k - \mathbf{s}_k^T\mathbf{y}_k} & \text{if } \mathbf{s}_k^T\mathbf{y}_k < 0.2\mathbf{s}_k^T\mathbf{B}_k\mathbf{s}_k \end{cases} \tag{3.23}$$

Algorithm 2 Hessian matrix update (BFGS update)

Input: $\mathbf{B}_k$

Output: $\mathbf{B}_{k+1}$

1: $\mathbf{s}_k \leftarrow \mathbf{x}_{k+1} - \mathbf{x}_k$

2: $\mathbf{y}_k \leftarrow \nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}_{k+1}, \boldsymbol{\lambda}_{k+1}) - \nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}_k, \boldsymbol{\lambda}_{k+1})$

3: $\mathbf{r}_k \leftarrow \theta_k\mathbf{y}_k + (1-\theta)\mathbf{B}_k\mathbf{s}_k$ with θ_k is defined as (3.23)

4: $\mathbf{B}_{k+1} \leftarrow \mathbf{B}_k - \dfrac{\mathbf{B}_k\mathbf{s}_k\mathbf{s}_k^T\mathbf{B}_k}{\mathbf{s}_k^T\mathbf{B}_k\mathbf{s}_k} + \dfrac{\mathbf{r}_k\mathbf{r}_k^T}{\mathbf{s}_k^T\mathbf{r}_k}$

3.3.2 Merit functions

A merit function is used to decide whether a trial step should be accepted in SQP method. For computing the step-size, inequality constraints $c(\mathbf{x}) \leq 0$ are often converted to the equality constraints with slack variables $s \leq 0$ as $\bar{c}(\mathbf{x}, s) = c(\mathbf{x}) - s$.

The ℓ_1 merit function takes the form:

$$\phi_1(\mathbf{x}; \mu) = f(\mathbf{x}) + \mu\|\mathbf{c}(\mathbf{x})\|_1 \tag{3.24}$$

We choose the penalty parameter μ large enough that:

$$\mu \geq \frac{\nabla_{\mathbf{x}}f_k^T\mathbf{p}_k + (\sigma/2)\mathbf{p}_k^T\nabla^2_{\mathbf{xx}}\mathcal{L}_k\mathbf{p}_k}{(1-\rho)\|\mathbf{c}_k\|_1} \tag{3.25}$$

where $\rho \in (0, 1)$ and:

$$\sigma = \begin{cases} 1 & \text{if } \mathbf{p}_k^T\nabla^2_{\mathbf{xx}}\mathcal{L}_k\mathbf{p}_k > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.26}$$

3.3.3 SQP algorithm

A general algorithm for SQP method is presented in Algorithm 3

Algorithm 3 SQP algorithm [Nocedal 2006]

Choose parameters: initial pair $(\mathbf{x}_0, \boldsymbol{\lambda}_0)$, $\eta \in (0, 0.5)$, $\tau \in (0, 1)$
Evaluate $f_0, \nabla f_0, \mathbf{c}_0, \mathbf{A}_0 = \nabla_{\mathbf{x}} \mathbf{c}_0$
Initialize symmetric positive definite Hessian approximation $\mathbf{B}_0$
while convergence conditions are not satisfied **do**
    Compute direction $\mathbf{p}_k$ by solving (3.20) using Algorithm 1; let $\hat{\boldsymbol{\lambda}}$ be the corresponding Lagrangian multipliers
    Set $\mathbf{p}_\lambda \leftarrow \hat{\boldsymbol{\lambda}} - \boldsymbol{\lambda}_k$
    Choose μ_k to satisfy (3.25) with $\sigma = 1$
    Set $\alpha_k \leftarrow 1$
    while $\phi_1(\mathbf{x}_k + \alpha_k \mathbf{p}_k; \mu_k) > \phi_1(\mathbf{x}_k; \mu_k) + \eta \alpha_k D_1(\phi(\mathbf{x}_k; \mu_k)\mathbf{p}_k)$ **do**
        Set $\alpha_k \leftarrow \tau_\alpha \alpha_k$ for some $\tau_\alpha \in (0, \tau]$
    end while
    Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$
    Set $\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \alpha_k \mathbf{p}_\lambda$
    Evaluate $f_{k+1}, \nabla f_{k+1}, \mathbf{c}_{k+1}, \mathbf{A}_{k+1} = \nabla_{\mathbf{x}} \mathbf{c}_{k+1}$
    Set $\mathbf{s}_k \leftarrow \alpha_k \mathbf{p}_k$ and $\mathbf{y}_k \leftarrow \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}_{k+1}, \boldsymbol{\lambda}_{k+1}) - \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}_k, \boldsymbol{\lambda}_{k+1})$
    Update $\mathbf{B}_{k+1}$ using Algorithm 2
end while

3.4 Multiobjective Optimization

3.4.1 Problem definition

In this section, we present popular methods of a MultiOjective Optimization (MOO) problem. Normally, MOO problem consists of conflicting objectives and there are many feasible solutions corresponding with chosen parameters. The final decision depends on the choice of decision maker (DM) (see Figure 3.1), which can be a software or a human. The classification of MOO methods is often based on the articulation of *preference information* in the solving process of MOO problem. If the preference information is used as a prior or a posterior information in mathematical programming solvers, the MOO methods are called non-interactive methods. Otherwise, if the preference information is progressively articulated in mathematical programming solvers, the MOO methods are called interactive methods. Other methods of MOO problem are heuristic approaches based on nature observations.

A general multiobjective minimization problem is defined by:

$$\begin{aligned} &\min_{\mathbf{x}} \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}) f_2(\mathbf{x}) \dots f_n(\mathbf{x})]^T \\ &s.t \quad \mathbf{x} \in \mathbb{X} \end{aligned} \tag{3.27}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the decision variable, $\mathbb{X}$ is the feasible set of the decision variable. $\mathbf{f}$ is the objective function vector or decision vector, and f_i is the i^{th} scalar objective function. Note that in multiobjective optimization problem, we have two important

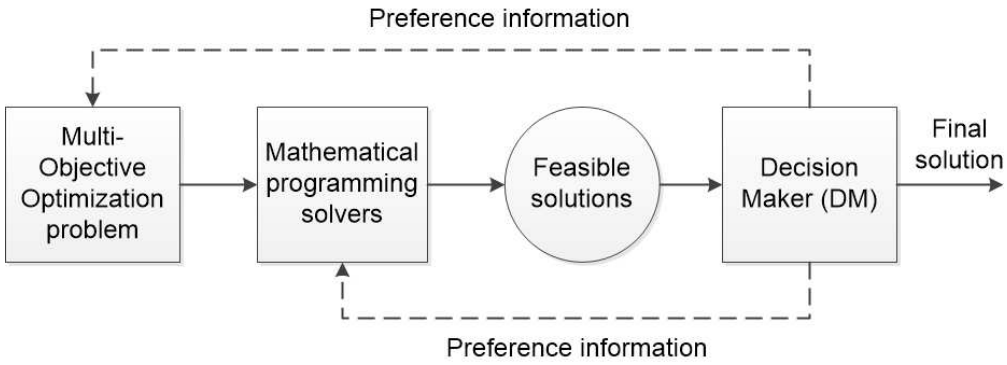


Figure 3.1 – Multiobjective optimization problem solving process

sets, namely decision variable set (feasible set) and decision vector set (feasible objective set). The convexity of each space is important to MOO problem.

3.4.2 Basic Definitions

This part presents fundamental definitions in multiobjective optimization problem which are based on [Chankong 2008] [Miettinen 1999]

Definition 4 *The multiobjective optimization problem is convex if all the objective functions and the feasible set are convex.*

Definition 5 *A decision vector* $\mathbf{z} \in \mathbb{R}^n$ *dominates a decision vector* $\mathbf{u} \in \mathbb{R}^n$ *if* $z_i \leq u_i$ *for all* $i = 1, 2, 3, ...n$ *and* $z_j < u_j$ *for at least one* $j \in \{1, 2, 3..., n\}$, $j \neq i$.

Definition 6 *A decision variable* $\mathbf{x} \in \mathbb{X}$ *with* $\mathbf{f}(\mathbf{x})$ *dominates a decision variable* $\mathbf{y} \in \mathbb{X}$ *with* $\mathbf{f}(\mathbf{y})$ *if* $f_i(\mathbf{x}) \leq f_i(\mathbf{y})$ *for all* $i = 1, 2, 3, ...n$ *and* $f_j(\mathbf{x}) < f_j(\mathbf{y})$ *for at least one* $j \in \{1, 2, 3..., n\}$, $j \neq i$.

Definition 7 *A decision variable* $\mathbf{x}^* \in \mathbb{X}$ *with* $\mathbf{f}(\mathbf{x})$ *is called (global) Pareto optimal (efficient, non-dominated, non-inferior) if and only if there exist no decision variable* $\mathbf{x} \in \mathbb{X}$ *such that* $f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)$ *for all* $i = 1, 2, 3, ...n$ *and* $f_j(\mathbf{x}) < f_j(\mathbf{x}^*)$ *for at least one* $j \in \{1, 2, 3..., n\}$ $j \neq i$.

Definition 8 *A decision variable* $\mathbf{x}^* \in \mathbb{X}$ *with* $\mathbf{f}(\mathbf{x})$ *is called weakly Pareto optimal if and only if there exist no decision variable* $\mathbf{x} \in \mathbb{X}$ *such that* $f_i(\mathbf{x}) < f_i(\mathbf{x}^*)$ *for all* $i = 1, 2, 3, ...n$.

Definition 9 *A decision variable* $\mathbf{x}^* \in \mathbb{X}$ *with* $\mathbf{f}(\mathbf{x})$ *is called locally Pareto optimal if and only if there exist* $\delta > 0$ *such that* $\mathbf{x}^*$ *is Pareto optimal in* $\mathbb{X} \cap \mathbf{B}(\mathbf{x}^*, \delta)$, *where* $\mathbf{B}(\mathbf{x}^*, \delta)$ *is an open ball of radius* δ *centered at* $\mathbf{x}^* \in \mathbb{X}$, *that is,* $\mathbf{B}(\mathbf{x}^*, \delta) = \{\mathbf{x} \in \mathbb{R}^d, \|\mathbf{x} - \mathbf{x}^*\| < \delta\}$.

Definition 10 *A decision variable* $\mathbf{x}^* \in \mathbb{X}$ *with* $\mathbf{f}(\mathbf{x})$ *is called properly Pareto optimal if it is Pareto optima and if there is some real number* $M > 0$ *such that for each* f_i *and each* $\mathbf{x} \in \mathbb{X}$ *satisfying* $f_i(\mathbf{x}) < f_i(\mathbf{x}^*)$*, there exists at least one* f_j *such that* $f_j(\mathbf{x}^*) < f_j(\mathbf{x})$ *and* $\frac{f_i(\mathbf{x}^*)-f_i(\mathbf{x})}{f_j(\mathbf{x})-f_j(\mathbf{x}^*)} \leq M$*.*

Definition 11 *The nadir decision vector is defined as* $\mathbf{z}^{nad} = (z_1, z_2, z_3, ..., z_n)^T$ *with elements* $z_i = \sup_{x \in \mathbf{P}_D} f_i(\mathbf{x}), i = 1, 2, ..., n$ *where* $\mathbf{P}_D$ *is the Pareto optimal decision variable set.*

Definition 12 *The ideal decision vector is defined as* $\mathbf{z}^* = (z_1, z_2, z_3, ..., z_n)^T$ *with elements* $z_i = \inf_{x \in \mathbf{P}_D} f_i(\mathbf{x}), i = 1, 2, ..., n$ *where* $\mathbf{P}_D$ *is the Pareto optimal decision variable set.*

Definition 13 *The utopian decision vector is defined as* $\mathbf{z}^{**} = (z_1, z_2, z_3, ..., z_n)^T$ *with elements* $z_i^{**} = z_i^* - \varepsilon$*, for all* $\varepsilon > 0, i = 1, 2, ..., n$*.*

Definition 14 *The aspiration level vector is defined as the decision vector that is satisfactory or desirable to the decision maker and denoted as* $\overline{\mathbf{z}} = (\overline{z}_1, \overline{z}_2, \overline{z}_3, ..., \overline{z}_n)^T$

Note that every global Pareto optimal decision variable is local Pareto optimal one but the converse is not always true. Every globally Pareto optimal decision variable is also weakly Pareto optimal. The illustration of Pareto front is displayed in Figure 3.2.

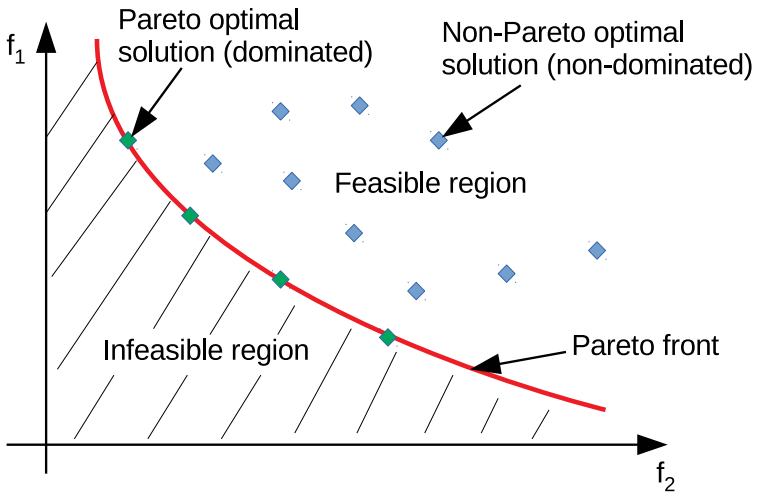


Figure 3.2 – Pareto front concept

3.5 Multiobjective optimization Methods

3.5.1 Non-interactive approaches

Non-interactive methods are not really multiobjective optimization. Indeed, the problem is converted into single objective optimization problem in different ways.

The simplest way is to use weighted sum. Another one is to solve the problem with one objective function while others are considered as nonlinear constraints. If we know the goal of each objective, goal programming or goal attaining technique is utilized. With these methods, a single solution is found if the problem is convex (all objectives are convex functions and the feasible set is a convex set). There are many algorithms for solving convex single objective optimization problem [Boyd 2004].

3.5.1.1 Weighting method

In the weighting method, we convert a MOO problem into a single objective problem by weighting each objective function. Particularly, with the weighting method, the original MOO problem can be casted as the following:

$$\begin{aligned} &\min_{\mathbf{x}} \sum_{i=1}^{n} \omega_i f_i(\mathbf{x}) \\ &s.t \quad \mathbf{x} \in \mathbb{X} \end{aligned} \tag{3.28}$$

where $\mathbf{x}$ is the decision variable vector, $\mathbb{X}$ is the feasible set, $f_i, i = \overline{1...n}$ is the i^{th} objective function, and ω_i is the weight corresponding with i^{th} objective function, and $\omega_i \geq 0$, $\sum_{i=1}^{n} \omega_i = 1$.

The solution of (3.28) can be proven to be weakly Pareto optimal, and strictly speaking, Pareto optimal if $\omega_i > 0$, for all $i = 1, 2..., n$, or if the solution is unique [Miettinen 1999], [Chankong 2008]. Then, if the problem (3.28) is convex, one unique solution can be found and this solution can be proven to be global Pareto optimal. However, the convexity of the problem is not easy to verify for many real applications. Note that the magnitude of objectives functions will affect in solving process and they should be normalized into dimensionless scale before applying the weighting method.

3.5.1.2 ε-constraint method

In the ε-constraint method, only one of the objective functions is selected to be optimized, the others are embedded into constraints. Therefore, the original MOO problem becomes:

$$\begin{aligned} &\min_{\mathbf{x}} f_k(\mathbf{x}) \\ &s.t \quad \mathbf{x} \in \mathbb{X} \\ &\qquad f_j(\mathbf{x}) \leq \varepsilon_j, \quad j = 1, ..., n, \quad j \neq k \end{aligned} \tag{3.29}$$

where $k \in \{1, 2, 3, ...n\}$ and ε_j are upper bounds for the objective functions, $j \neq k$.

It is clear that this is also a single objective function optimization problem. The preference information of decision maker is the upper bounds of objective functions. Regarding the optimality, a solution of problem (3.29) is proven to always be weakly

Pareto optimal [Chankong 2008]. Moreover, the unique solution of (3.29) is Pareto optimal for any given upper bound ε_j. Theoretically, this method can find all Pareto optimal solutions by changing the upper bounds of objective functions even though feasible objective set is non-convex. However, it is not easy to choose these upper bounds properly.

3.5.1.3 Global criterion method

In global criterion approach, the distance between a desirable point and current point is optimized. The problem is changed to:

$$\begin{aligned} \min_{\mathbf{x}} & \sum_{i=1}^{n} \|f_i(\mathbf{x}) - z_i^\star\|_p \\ s.t & \quad \mathbf{x} \in \mathbb{X} \end{aligned} \tag{3.30}$$

where $z_i^\star$ is the desirable reference point or ideal decision vector which is selected by the DM. Note that if the objective functions have different magnitudes, this method only works properly if we normalize the problem into uniform, dimensionless scale and the choice of the norm-p affects to the results of the problem. We can prove that a solution of (3.30) is Pareto optimal.

3.5.1.4 Neutral compromise method

In neutral compromise approach, the problem to solve is:

$$\begin{aligned} \min_{\mathbf{x}} & \quad \max_{i=1,\ldots,n} \frac{f_i(\mathbf{x}) - ((z_i^\star + z_i^{nad})/2}{z_i^{nad} - z_i^{\star\star}} \\ s.t & \quad \mathbf{x} \in \mathbb{X} \end{aligned} \tag{3.31}$$

where $z_i^\star$ is the ideal objective point, z_i^{nad} is the nadir objective point, $z_i^{\star\star}$ is the utopian objective point.

As can be seen, the nadir and ideal objective point are used for scaling purposes. The solution of (3.31) is proven to be weakly Pareto optimal [Chankong 2008].

3.5.1.5 Value function method

In the value function method, a value function is defined and maximized as:

$$\begin{aligned} \max_{\mathbf{x}} & \quad \nu(\mathbf{f}(\mathbf{x})) \\ s.t & \quad \mathbf{x} \in \mathbb{X} \end{aligned} \tag{3.32}$$

where $\nu(.)$ is the value function representing the opinions of the DM.

This method is likely simple to solve (single objective optimization and no additional constraints in feasible set). Nevertheless, the challenging issue of the method is how to build the appropriate value function that covers all the preference information and easy to solve, for instance, it should be convex function. The solution of (3.32) is Pareto optimal if the value function is strongly decreasing [Miettinen 1999].

3.5.1.6 Lexicographic ordering method

In lexicographic ordering method, the objective functions are arranged according to their absolute importance. Firstly, the most important objective function is minimized subject to the original constraints. If this problem has a unique solution, the solution process stops. Otherwise, the second most important is minimized and the new constraints are introduced in order to guarantee the optimality of the most important objective function. The process is repeated. However, it is not trivial to get the unique solution unless solving problem is convex or in the special case. The solution of lexicographic ordering method can be proven to be Pareto optimal. Two challenging questions arise that the difficulties of DM to specify the absolute importance of objective functions and some objective functions do not take into account in solution process if this process stops before the less important objective function taken into consideration [Miettinen 1999] [Chankong 2008].

3.5.1.7 Global programming method

In the global programming method, the original MOO problem is rewritten:

$$\begin{aligned} \min_{\mathbf{x}} & \sum_{i=1}^{N} \omega_i \delta_i \\ s.t \quad & f_i(\mathbf{x}) - \delta_i \leq \overline{z}_i, \quad i = 1, ..., n \\ & \delta_i \geq 0, \quad i = 1, ..., N \\ & \mathbf{x} \in \mathbb{X} \end{aligned} \tag{3.33}$$

where $\mathbf{x}$ and δ_i are the variables, $\overline{z}_i$ is the aspiration level of objective vector. There are several versions of global programming methods such as lexicographic goal programming approach, min-max goal programming method, and meta-goal programming. However, the underlying philosophy of all goal programming methods is to minimize the deviation of objective function vector and its aspiration level vector. In virtue of many versions of global programming method, the necessary conditions of Pareto optimal solutions of each method can be found in [Miettinen 1999] and references therein.

3.5.1.8 Weighted metrics method

The method generalizes the global criterion method, in the weighted metrics method, the problem is that:

$$\begin{aligned} \min_{\mathbf{x}} & \sum_{i=1}^{n} \omega_i (\| f_i(\mathbf{x}) - z^{\star} \|_p) \\ s.t \quad & \mathbf{x} \in \mathbb{X} \end{aligned} \tag{3.34}$$

where z^* is the ideal objective point. Regarding the optimality, the solution of (3.34) is proven to be Pareto optimal if either the solution is unique or all the weights are

positive [Miettinen 1999]. Moreover, if the problem is convex, all Pareto optimal solutions can be found by altering the weights.

3.5.1.9 Achievement scalarizing function method

In achievement scalarizing function approach, the MOO problem is casted as:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \max_{i=1,\dots,n} [\omega_i (f_i(\mathbf{x}) - \overline{\mathbf{z}}_i] + \rho \sum_{i=1}^{n} (f_i(\mathbf{x}) - \overline{\mathbf{z}}_i) \\ s.t \quad & \mathbf{x} \in \mathbb{X} \end{aligned} \tag{3.35}$$

where $\overline{\mathbf{z}}$ is the reference point, ω_i is the normalizing factor, and $\rho > 0$ is an augmentation multiplier. The solution of problem (3.35) can be proved properly Pareto optimal and any properly Pareto optimal solution can be found.

3.5.1.10 Goal attainment method

This method was first introduced in [Gembicki 1975]. The MOO problem is written as:

$$\begin{aligned} & \min_{\mathbf{x},\gamma} \gamma \\ s.t \quad & \mathbf{x} \in \mathbb{X} \\ & \mathbf{f}(\mathbf{x}) - \mathbf{w}\gamma \leq \bar{\mathbf{f}} \end{aligned} \tag{3.36}$$

where $\mathbf{w}$ is attainment vector and $\bar{\mathbf{f}}$ is desirable objective vector (goal vector). The quantity $\mathbf{w}\gamma$ is related to the degree of under- or over-attainment of the goal $\bar{\mathbf{f}}$. Different Pareto optimal solutions with different degree of under-attainment can be found.

3.5.1.11 Approximation methods

In the approximation methods, strategies to approximate the set of Pareto optimal solutions are introduced. A survey of these methods can be found in [Ruzika 2005].

3.5.2 Interactive approaches

Interactive approaches are the approaches in which the decision maker provides preference information in a repeated step of an interactive algorithm. There are three types of specifying preference information based on trade-off information, reference points, and classification of objective functions.

3.5.2.1 Trade-off based methods

In trade-off based methods, some trade-off concepts have been introduced, namely *objective trade-off* and *subjective trade-off*. Objective trade-off is a measurement of

the change in one objective function relating to the change in another one when the variable moves from one feasible point to another one. Subjective trade-off is a measurement of how much the desirable sacrifice in the value of some objective functions in order to improve another objective function to a certain quality. Two these trade-off definitions are integrated in interactive methods to decide the next Pareto optimal solution. Several trade-off based methods were introduced in literature such as Z-W (Ziont-Wallenius) method, ISWT (Interactive Surrogate Worth Trade-off) method, GDF (Geoffrion-Dyer-Feinberg) method, SPOT (Sequential Proxy Optimization Technique) method, and GRIST (GRadient based Interactive Step Trade-off) method.

3.5.2.2 Reference point methods

The fundamental philosophy of reference point methods is that the decision maker should learn during the interaction with a Decision Support System (DSS), an interaction system between an mathematical algorithm and decision maker. Firstly, the DM give reference points to DSS, DSS responds by maximizing the achievement function and the DM can be free to modify reference points. This leads to another aspect, learning perspectives of interactive multi-optimization methods. There are two general learning methods, namely individual learning and model learning.

3.5.2.3 Classification-based methods

Three popular classification-based methods are STEM method (STep Method), STOM method (Satisficing Trade-Off Method), and NIMBUS method (Non differentiable Interactive Multi-objective BUndle-based optimization System). They differ in the classification way, the preference information asked from the Decision maker, and how to generate a new Pareto optimal solution.

This section briefly presents interactive methods in MOO problem. Further details of these methods can be found in [Branke 2008] and references therein.

Note that, as can be seen, a MOO problem is converted to a single optimization problem in different ways in all classical methods (non-interactive and interactive methods). Then, the single optimization solvers play a pivotal role in finding the solutions of MOO. In practice, numerical solvers of single optimization problem only find local optimal solutions except for the problem is convex or in a special case. Therefore, just local Pareto optimal solutions can be expected to be found in general.

3.5.3 Heuristic approaches

Heuristic approaches include evolutionary methods and swarm-based methods (particle swarm, ant colony). These methods are based on the natural process, or insects and animals observation. The famous and first evolutionary algorithm is the genetic algorithm [Holland 1975]. There are some extended genetic algorithm for MOO problem, namely vector evaluated genetic algorithm (VEGA)

[Schaffer 1985], multi-objective genetic algorithm (MOGA) [Fonseca 1993], non-dominated genetic algorithm (NSGA) [Srinivas 1994], Niched Pareto genetic algorithm (NPGA) [Horn 1994], Non-dominated genetic algorithm II (NSGA-II) as in [Deb 2002]. With the rapid development of biological-mathematical studies in recent years, a lot of evolutionary algorithms has been presented such as Strength-Pareto evolutionary algorithm (SPEA) [Zitzler 1999, Zitzler 2001], multi-objective particle swarm optimization (MOPSO) [Coello 2004], Multi-objective evolutionary algorithm based on decomposition (MOEA/D) [Zhang 2007], Pareto archived evolution strategy (PAES) [Knowles 2000], Pareto-frontier differential evolution (PDE) [Abbass 2001], and multi-objective Grey wolf (MOGO) [Mirjalili 2016]. The main difference and advantage of heuristic methods comparing to classical approaches (non-interactive and interactive methods) is that multiple trade-off solutions can be found in a single simulation run. Moreover, the objective functions can not be continuous. Because evolutionary and swarm based methods are heuristic based procedure, they are not guarantee in finding Pareto optimal solutions as provable optimization methods would do. However, these algorithms have essential operators to constantly improve the evolving non-dominated points [Branke 2008]. One more drawback of the heuristic approaches is not easy to consider nonlinear constraints in solving process. The fundamental philosophy for an evolutionary algorithm is as in Table 3.1 where $\mathbf{P}_t$ is the population at step t, $\mathbf{P}_t^{'}$ and $\mathbf{P}_t^{''}$ are the populations corresponding with selection and variation operator.

$t = 0$
Initialization($\mathbf{P}_t$)
do
Evaluation($\mathbf{P}_t$)
$\mathbf{P}_t^{'}$=Selection($\mathbf{P}_t$)
$\mathbf{P}_t^{''}$=Variation($\mathbf{P}_t^{'}$)
$\mathbf{P}_{t+1}$ = Elitism($\mathbf{P}_t$,$\mathbf{P}_t^{''}$)
$t = t + 1$
While
Termination ($\mathbf{P}_t$,$\mathbf{P}_{t+1}$)

Table 3.1 – Basic evolutionary algorithm [Branke 2008]

3.6 Conclusion

In general, optimization algorithms are the key of many strategies and applications such as model predictive control, communication, operations research, and machine learning. This chapter presented basic definitions and methods in single-objective optimization and multiobjective optimization problems. In particular, for nonlinear programming, sequential quadratic programming based on active-set method, which is the basic of proposed approach in the chapter 7 and suitable for medium problem (otherwise, interior-point method is more efficient for large

problem), was described. SQP method sequentially solves equality quadratic programming with active-set, which is changed in each step or not, depending on corresponding Lagrange multipliers. Many approaches for multiobjective optimization were reviewed from deterministic to heuristic directions in which one method was chosen to solve problem in the chapter 4. Heuristic methods, based on nature rules such as genetic, particle swarm, Grey Wolf algorithms and their variants, can find a global optimal value but it can not be proved. For deterministic methods (weighting method, ε constraint method, global criterion method, neutral compromise method, and so on), it is easy to stuck in local optimal value and its global solution depends the convexity of objective functions. It is clear that most of multiobjective optimization methods deploy techniques in one-objective optimization. The choice of which methods depends on the properties of objective functions and constraints. In multiobjective optimization domain, concept and related definitions of Pareto solution were described. For continuous space, finding all Pareto points is a challenging problem. The question of the choice of a suitable Pareto point has to done by decision maker.

Chapter 4

Performance Indices and Static Configuration Design

Contents

This chapter presents static configuration design problem for over-actuated underwater robots. Performance criteria, including manipulability, energetic, workspace, reactive, and robustness indices, are proposed and analyzed. A multiobjecitve optimization method is chosen and a procedure of searching optimal solution is proposed.

4.1 Introduction

The Actuation System (AS) groups the different actuators carried by the system. Following the generic Navigation-Guidance-Control (NGC) control structure, the AS is in charge of realizing the desired force vector ($\mathbf{F}_B^d$) provided by the control system (see Fig 4.1). Following Figure 4.1, the Sensorial Stage uses sensors measurement and prior knowledge of the environment to provide to the navigation system the necessary information to compute an estimation of system state ($\hat{\boldsymbol{\eta}}$). Then the guidance system uses this estimation and the reference system state ($\boldsymbol{\eta}^d$) provided by the mission controller to compute the error function (ε). The control system is then in charge of computing the desired force ($\mathbf{F}_B^d$) in order to reduce the error function to zero. Note that classically this desired force is expressed in

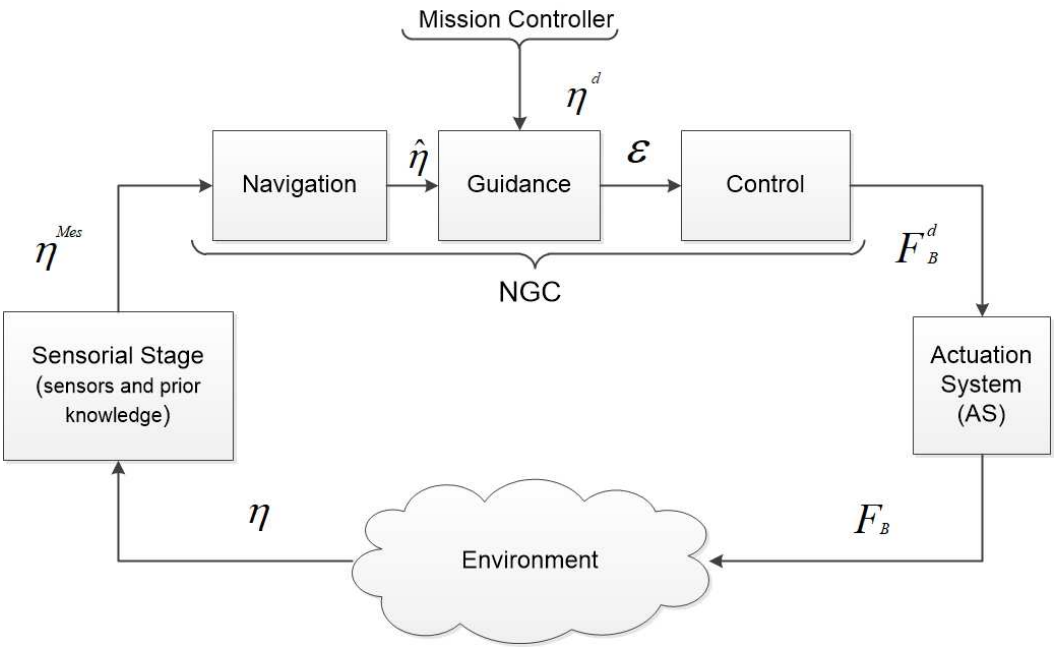


Figure 4.1 – NGC structure augmented with the Actuation System and Sensorial Stage [Dang 2019]

the body frame. Afterwards, the *Actuation system* produces on the environment a resulting force vector ($\mathbf{F}_B$), which should be as close as possible to $\mathbf{F}_B^d$. Note that, in this thesis, desired force ($\mathbf{F}_B^d$) and resulting force ($\mathbf{F}_B$) are (6×1) vectors and include force and torque elements.

Referring to Figure 4.2, the desired force ($\mathbf{F}_B^d$) is the output of the controller. Then the Dispatcher ($\mathcal{D}$) (see Chapter 2) considers actuator allocation method (and eventually redundant management) to compute the desired actuators force ($\mathbf{F}_m^d$) that each actuator has to produce. The inverse actuators characteristics are then considered in order to compute the actuators input ($\mathbf{c}_m$). Once applied, $\mathbf{c}_m$ produces actuators forces ($\mathbf{F}_m$). The resulting force $\mathbf{F}_B$ is produced with respect to the actuators configuration (modeled as $\mathbf{A}$ matrix, called *configuration matrix*)(see Chapter 2).

The properties of the AS are indeed dependent of the actuators configuration (position and attitude with respect to the body frame), actuators dynamics (response characteristics), and Dispatcher (control allocation, redundancy management) (see Fig 4.2), and afford the system with different properties. Let's consider in the following that n is the number of DOFs of the system, and m is the number of actuators. If the system carries less actuators than Degrees of Freedom (DOFs), it is said to be *under-actuated* (in that case, $\mathbf{A}$ will be a ($n \times m$) matrix where $n > m$). Long-range autonomous underwater vehicles (AUVs) and, for the terrestrial case, unicycle wheeled vehicles belong this category [Lapierre 2009]. In that case, specific nonlinear guidance strategies have to be used [Lapierre 2008]. If the

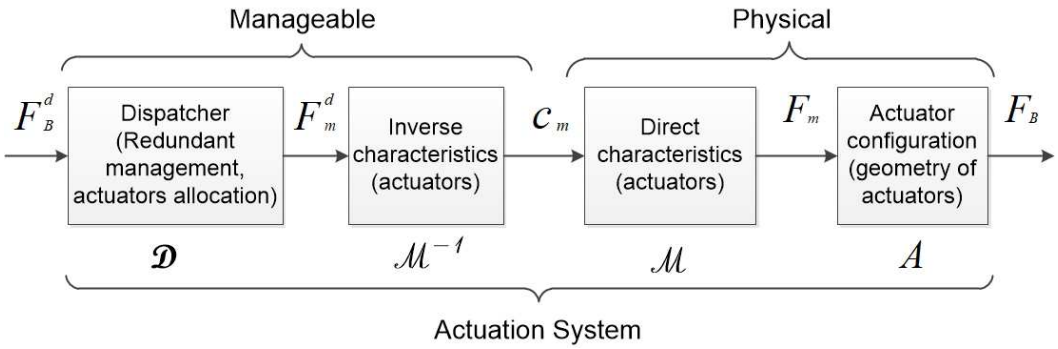


Figure 4.2 – Actuation system scheme

system carries more actuators than DOFs, it is said to be *redundant* ($n < m$). Then there are different solutions ($\mathbf{c}_m$) to produce an identical resulting force ($\mathbf{F}_B$). Indeed, $\mathcal{D}$ is one of the multiple possible inverses of $\mathbf{A}$, classically, $\mathcal{D} = \mathbf{A}^+$ where $\mathbf{A}^+$ is the Moore-Penrose pseudo-inverse. Note that this property of redundancy does not systematically afford the system with *over* (or even *iso*)-actuated (fully actuated) condition. Indeed, redundancy has to be studied along each DOF and a redundant system can be under-actuated if some actuators act along the same DOF or deliver their action along the system's center of mass. But if the AS is configured in order to have an action on all the DOFs, the system is said to be *iso-actuated* (if it carries as many actuators as DOFs, then in this case $\mathbf{A}$ is a square matrix ($n \times n$)), or *over-actuated* (if it carries more actuators than DOFs). In that case, linearized approaches (for small velocities) hold [Fossen 2011]. The properties of the AS plays a pivotal role in the system performances, in terms of achievable dynamics, manoeuvrability, robustness and dependability.

The properties of an *over-actuated* system have been studied in aerospace control, where critical safety is required in [Levine 2010], and for marine vehicles in [Johansen 2013], where the harsh oceanic conditions may easily produce actuator failure. Redundancy has also been used in [Ropars 2018] in order to compensate different and unknown actuators responses. The domain of robotic manipulator has also extensively studied this question of redundancy; especially with recent works on humanoid robotics, where task function approach [Nakamura 1987] has been used to achieve concurrently equilibriums [Adorno 2010], walking pattern following [Agravante 2016] and multi-contact management [Pham 2018].

For a global evaluation of an *Actuation System*, we should of course consider many factors, including redundancy management, control allocation method, actuator characteristics (inverse and direct), and actuators configuration. This thesis focuses on the study of actuators configuration, other problems can be referred to [Ropars 2018] and references therein.

Different performance criteria related to the actuators configuration design have been proposed. For mobile manipulation, *manipulability index* [Yoshikawa 1985a] measures the manipulation capability of the end-effector. Intuitively, this index

points the set of all end-effector velocities which is realizable by joint velocities. This set is called manipulability ellipsoid. This index is quantified by computing manipulability ellipsoid properties. Based on these properties, there are different ways to quantify the manipulability index, including the volume of manipulability ellipsoid, the ratio of the minimum and maximum radii of the hyper-ellipsoid, the minimum radius of the hyper-ellipsoid. When the uniformity of manipulating ability is important, the ratio of two radii of the hyper-ellipsoid is chosen (optimal value will be closed to 1). Otherwise, the minimum radius of the hyper-ellipsoid is suited for the case where the minimum manipulating ability might be critical [Yoshikawa 1990]. *Attainability* [Kumar 1981], [Paden 1988], [Park 1994] was studied using workspace volume estimation.

In the marine robotic field, *manipulability index*, *energetic index*, and *force index* were introduced in [Pierrot 1998] and manipulability index was applied in [Kharrat 2015]. Specifically, the *manipulability index* is used to measure the system's ability to exert a desired force with a specific actuator configuration. So, the closer to 1 this index is, better the robot isotropy is, i.e, the robot can exert the same forces/torques in any directions. The *energetic index* is a measurement of the variation of system energy when the direction of desired force changes. This is realized by a measurement of energy consumption when the direction of an unit desired force changes all over a 3D sphere. The basic idea of energetic index is to keep system's energy consumption constant and as low as possible when the direction of action changes. The *force index* is used to measure the ratio between actual maximum and minimum realizing forces. However, these studies only consider a given and fixed actuators configuration.

Regarding the design of actuators configuration of an over-actuated system in marine robotics, a general problem is: how to achieve an optimal configuration that considers different indices. This raises two specific questions:

1. how to define general and typical indices to evaluate an actuators configuration of an over-actuated marine system.
2. how to solve the complex optimal problem, which is normally non-convex and has some conflicting objectives

This thesis focuses on the design of an actuators configuration of an over-actuated marine robot which optimizes different performance indices. Mathematically, an actuators configuration is a mapping between an actuators force vector and a resulting force vector. Since we are considering a marine system equipped with thrusters, the mapping will be from a thrusters force vector ($\mathbf{F}_m$ space) to a body frame force vector ($\mathbf{F}_B$ space) with a relation $\mathbf{F}_B = \mathbf{A}\mathbf{F}_m$, (see Figure 4.3). The mapping is a matrix with some names in the literature such as: control effectiveness matrix [Johansen 2013], [Stephan 2017] static transformation matrix [Grechi 2016], geometrical distribution of thrusters [Ropars 2015], configuration matrix [Kharrat 2015]. In this thesis, a mapping of an actuators configuration is called a *configuration matrix*, $\mathbf{A}$.

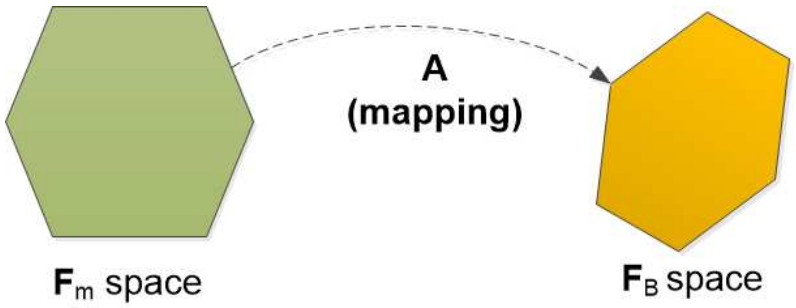


Figure 4.3 – Actuators configuration mapping

4.2 Problem formulation

The relation of desired force ($\mathbf{F}_B^d$) and resulting force ($\mathbf{F}_B$) depends on different factors (see Figure 4.2). This paper only focuses on actuators configuration. Therefore, three assumptions are outlined below:

1. *Inverse characteristics and direct characteristics of actuators are perfectly known, i.e,* $\mathbf{F}_m^d = \mathbf{F}_m$
2. *Dispatcher is the Moore-Penrose pseudo-inverse of actuators configuration, i.e, if actuators configuration is* $\mathbf{A}$ *matrix, dispatcher is* $\mathcal{D} = \mathbf{A}^+$
3. *All actuators have the same characteristics*

4.2.1 Model of actuators configuration

This part describes how to model an actuators configuration of an over-actuated marine robot equipped with thrusters. A thruster is modelled by its position and direction with respect to body frame of the robot. The position of the i^{th} thruster is described by an unit position vector $\mathbf{r}_i$ and distance d_i to center of mass (CM) in the body frame. The direction of i^{th} thruster is represented by an unit vector direction $\mathbf{u}_i$ with respect to the body frame as in Fig 4.4, and the i^{th} thruster propels a force with magnitude of $F_{m,i}$. The relation of thruster forces vector and resulting forces vector (note that this space includes forces ($\mathbf{f}$) and torques ($\mathbf{\Gamma}$) is described in Equation (4.1).

$$\mathbf{F}_B = \mathbf{A}\mathbf{F}_m = \begin{pmatrix} \mathbf{F} \\ \mathbf{\Gamma} \end{pmatrix} \tag{4.1}$$

where resulting vector $\mathbf{F}_B = [F_u \quad F_v \quad F_w \quad \Gamma_p \quad \Gamma_q \quad \Gamma_r]^T \in \mathbb{R}^6$, $\mathbf{A} \in \mathbb{R}^{6\times m}$, and thruster force vector $\mathbf{F}_m = [F_{m,1} \quad F_{m,2} \quad ... \quad F_{m,m}]^T \in \mathbb{R}^m$, and m is the number of thrusters, $m > 6$.

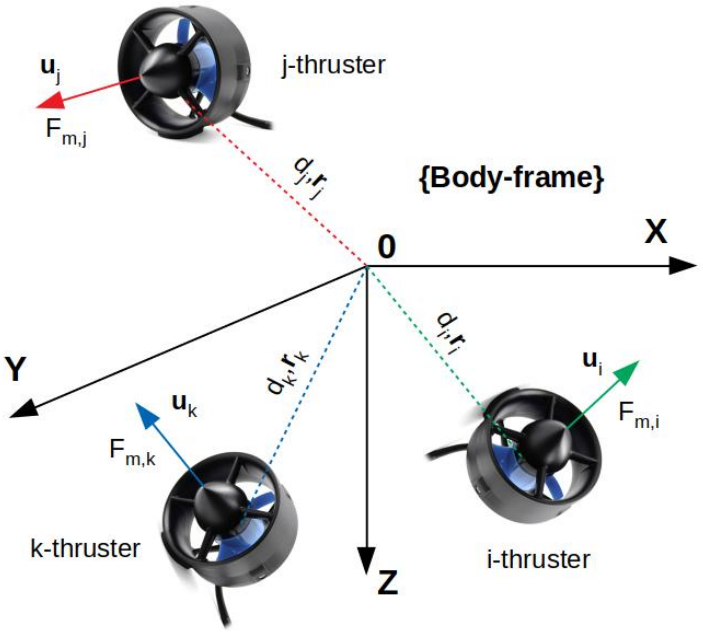


Figure 4.4 – Actuators configuration model

The configuration matrix $\mathbf{A}$ is described:

$$\begin{aligned}\mathbf{A} &= \begin{pmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_m \\ d_1\mathbf{r}_1 \otimes \mathbf{u}_1 & d_2\mathbf{r}_2 \otimes \mathbf{u}_2 & \cdots & d_m\mathbf{r}_m \otimes \mathbf{u}_m \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_m \\ \boldsymbol{\tau}_1 & \boldsymbol{\tau}_2 & \cdots & \boldsymbol{\tau}_m \end{pmatrix} = \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{pmatrix}\end{aligned} \tag{4.2}$$

where $\mathbf{A}_1$, $\mathbf{A}_2 \in \mathbb{R}^{3\times m}$ are sub-matrices of $\mathbf{A}$ which involve force and torque elements respectively. It is obvious to see that $\boldsymbol{\tau}_i^T.\mathbf{u}_i = 0$. This is one of constraints of the configuration matrix structure.

In this chapter, we assume that all distances from thrusters positions to the center of body-frame are the same, $d_i = d_j = const, i, j = 1...m, i \neq j$. Without loss of generality, we can assume that $d_i = 1, i = 1, ..., m$.

4.3 The different indices

The preliminary condition for the following studies is that the rank of configuration matrix equals six ($rank(\mathbf{A}) = 6$). This section introduces performance criteria to evaluate a configuration of an underwater robot. They are manipulability, energetic, workspace, reactive, and robustness indices.

4.3.1 Manipulability index

As mentioned before, manipulability index was first introduced for manipulator mechanisms [Yoshikawa 1985b], and there are different ways to quantify the manipulability index. This paper focuses on the isotropy property of a marine robot. Then, the ratio of maximum and minimum radii of the manipulability ellipsoid is chosen (see Figure 4.5). Because of units consistency, the matrices which involve force space, $\mathbf{A}_1$, and torque space, $\mathbf{A}_2$, are investigated separately. However, because of our assumption of d_i, the manipulability index is defined as the condition number of the configuration matrix:

$$I_m = Cond(\mathbf{A}) = \frac{\sigma_{max}}{\sigma_{min}} \tag{4.3}$$

where σ_{max} and σ_{min} are the maximum and minimum singular value of configuration matrix, $\mathbf{A}$, respectively.

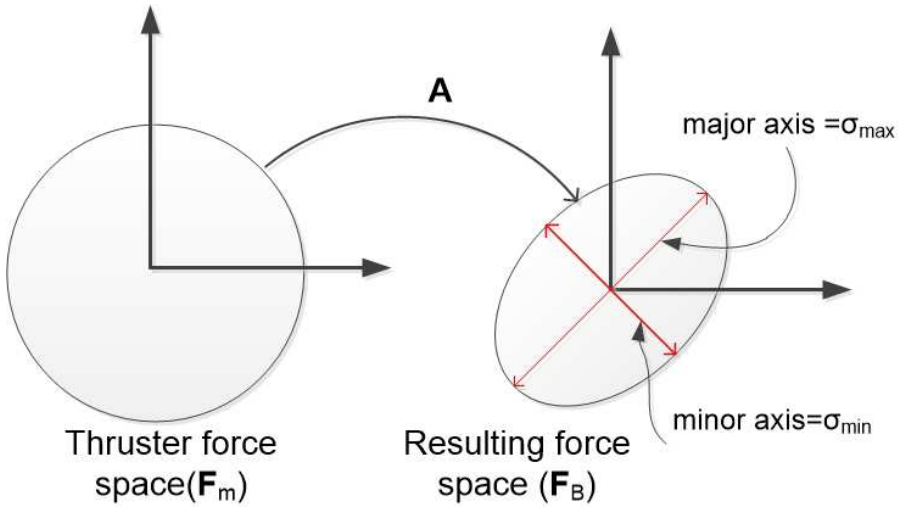


Figure 4.5 – Manipulability ellipsoid with mapping

Following Figure 4.5, manipulability index investigates the resulting force ellipsoid which is realizable by thrusters forces ($\mathbf{F}_m$) such that $\|\mathbf{F}_m\| \leq 1$ (see Theorem A.1.1 in Appendix). If $I_m = 1$, the robot is isotropic or if $I_m = \infty$, there is at least one direction in which the actuators are not acting and the rank of configuration matrix is less than six (6).

4.3.2 Energetic index

Energy is very important for underwater robots in ocean and karst exploration, and energy consumption of robots depends on a lot of factors such as mechanical designs, environmental effects, and a specific mission. In order to evaluate the energy performance of a marine robot, energetic index was introduced in [Pierrot 1998]. In this thesis, the norm of thruster force vector, $p_E = \|\mathbf{F}_m\|_2$, is used to qualify the

energy consumption that a marine robot uses to produce forces and torques, and can be calculated as in Equation (4.4).

$$p_E = \|\mathbf{F}_m\|_2 = \sqrt{\sum_{i=1}^{m} F_{mi}^2} = \|\mathbf{A}^+.\mathbf{F}_B^d\|_2 \tag{4.4}$$

The energetic index is proposed to measure the variation of energy consumption of a marine robot when the direction of desired force changes. It is quantified by computing the energy consumption when an unit desired force vector, ($\mathbf{F}_B^d$), changes all over hyper-sphere (see Figure 4.6 for 3D sphere). Because of units consistency, however, force and torque sphere are computed separately.

For the force sphere case, the unit desired force vector includes an unit vector of force elements and a zero vector of torque elements. For the torque sphere case, the unit desired force vector includes a zero vector of force elements and an unit vector of torque elements. Intuitively, this can be expressed as:

$$\mathbf{F}_B^d = \begin{pmatrix} \mathbf{F}^d \\ \mathbf{\Gamma}^d \end{pmatrix} = \begin{cases} \binom{\mathbf{u}_s}{\mathbf{0}}, & \text{for force sphere} \\ \binom{\mathbf{0}}{\mathbf{u}_s}, & \text{for torque sphere.} \end{cases} \tag{4.5}$$

where $\mathbf{u}_s = [\cos\alpha\cos\beta \quad \sin\alpha\cos\beta \quad \sin\alpha]^T$ is an unit vector in spherical coordinates with $\alpha \in [-\pi, \pi]$, and $\beta \in [-\pi/2, \pi/2]$.

According to two cases, the norm of thruster force vector is also divided into two cases as follows:

$$p_E = \begin{cases} p_{Ef} = \|\mathbf{A}^+\binom{\mathbf{u}_s}{\mathbf{0}}\|, & \text{for force sphere case} \\ p_{E\Gamma} = \|\mathbf{A}^+\binom{\mathbf{0}}{\mathbf{u}_s}\|, & \text{for torque sphere case.} \end{cases} \tag{4.6}$$

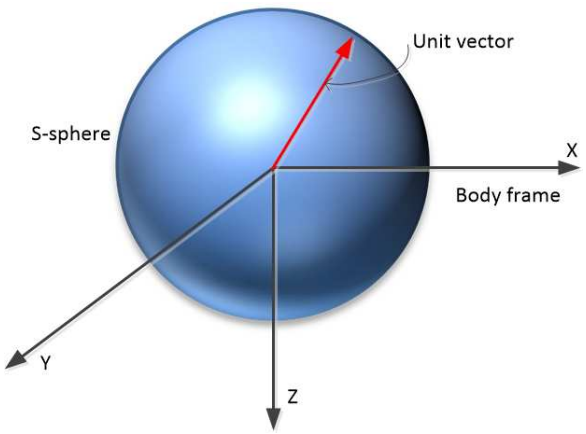


Figure 4.6 – The rotation of unit desired vector in 3D sphere

The energetic index is defined as:

$$I_e = \frac{1}{S}\int_S (w_{ef}p_{Ef} + w_{e\tau}p_{E\Gamma})dS \tag{4.7}$$

where S is the area of 3-dimensional sphere; p_{Ef}, $p_{E\Gamma}$ are the sub-vectors of p_E corresponding with force sphere and torque sphere case, respectively; and w_{ef} and $w_{e\Gamma}$ are weighting coefficients.

4.3.3 Workspace index

The term of workspace volume was first introduced in [Paden 1988] for manipulator systems. In this thesis, the workspace index is used to measure the volume of attainable regions of resulting force space w.r.t body frame. In general, characteristics of thrusters always have limitations, namely saturations and dead-zones (in this index, dead-zone is neglected). These yield to the polytope of thrusters force space, $\mathbf{F}_m$ space, denoted as $\mathbb{M}$. By properly choosing configuration matrix, $\mathbf{A} = (\mathbf{A}_1 \quad \mathbf{A}_2)^T$, the volume of the resulting force space for force, $\mathbb{F}_F$ space, and the resulting force space for torque, $\mathbb{F}_T$ space can be maximized (see Figure 4.7). Note that resulting spaces for force and torque are studied separately because of units consistency. In general, the set $\mathbb{M}$ of thrusters forces is known (given saturations),

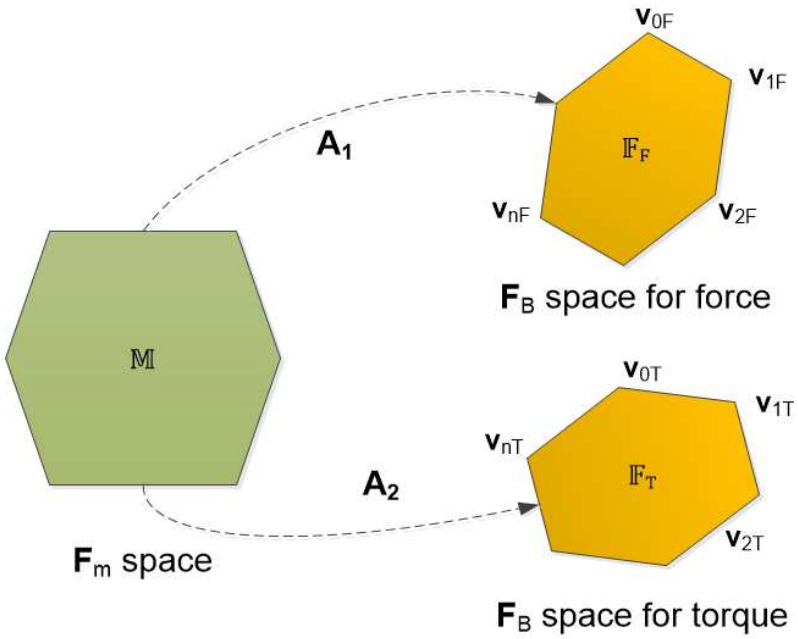


Figure 4.7 – Space Mapping

so $\mathbb{M}$ is a polytope and $\mathbb{F}_F$ and $\mathbb{F}_T$ are also polytopes through a linear transform, configuration matrix. We define the workspace index as

$$I_w = \omega_{wf} Vol(\mathbb{F}_F) + \omega_{w\tau} Vol(\mathbb{F}_T) \tag{4.8}$$

where Vol is the volume measure of a space, ω_{wf} and $\omega_{w\tau}$ are weighting coefficients.

In control perspectives, the larger spaces's volume are, the less control efforts are. The design objective is to maximize the workspace index, I_w. Normally, the

set $\mathbb{M}$ is convex and its vertices are known. It is easy to find the vertices of $\mathbb{F}_F$ and $\mathbb{F}_T$. Of course $\mathbb{F}_F$ and $\mathbb{F}_T$ are also convex sets (because of linear transformation). This problem becomes a volume computation of convex polytopes.

4.3.4 Reactive index

Reactive index quantifies how fast the actuation system is able to change the orientation of the resulting force $\mathbf{F}_B$ (ideally $\mathbf{F}_B^d$). Suppose that the robot is travelling in a direction with a set of thrusters forces $\mathbf{F}_{m1}$ induced from desired force vector $\mathbf{F}_{B1}^d$. The robot wants to change to another direction (or the same direction with the different manigtude) with the desired force vector $\mathbf{F}_{B2}^d$, so thrusters have to produce another set of thruster forces $\mathbf{F}_{m2}$. The 2-norm of deviation of thruster forces, $\triangle\mathbf{F}_m = \mathbf{F}_{m1} - \mathbf{F}_{m2} = [\triangle F_{m1} \triangle F_{m2} \cdots \triangle F_{mm}]^T$, is considered as the reactive capability of the robot. Referring to the approximation of characteristic of thrusters as Fig 4.8, the moving time from F_{m1} to F_{m2} is less than the moving time from F_{m1} to F_{m3} (in linear section, the dead-zone of thrusters charactersistics is neglected in this thesis). Hence, we have:

$$\triangle\mathbf{F}_m = \mathbf{A}^+(\mathbf{F}_{B1}^d - \mathbf{F}_{B2}^d) = \mathbf{A}^+\triangle\mathbf{F}_B^d \tag{4.9}$$

$$\|\triangle\mathbf{F}_m\| = \|\mathbf{A}^+\triangle\mathbf{F}_B^d\| \leq \|\mathbf{A}^+\|\|\triangle\mathbf{F}_B^d\| \tag{4.10}$$

$$\frac{\|\triangle\mathbf{F}_m\|}{\|\triangle\mathbf{F}_B^d\|} \leq \|\mathbf{A}^+\| \tag{4.11}$$

From Equation (4.11), the sensitivity of the thruster forces with respect to desired forces, in other words the variation of thruster forces w.r.t desired forces, is upper-bounded by the norm of pseudo-inverse of the configuration matrix, $\|\mathbf{A}^+\|$. We define the reactive index as:

$$I_{re} = \|\mathbf{A}^+\| \tag{4.12}$$

It is obvious to see that if this index is more less, the robot is more reactive. Then, the objective of design process is to minimize reactive index.

4.3.5 Robustness index

This criterion measures the robust level the AS of a marine robot. It means that if any thrusters of the robot fails, the remaining ones can still perform the robot's mission. In particular, for any $\mathbf{F}_B^d$ vector, there always exists a $\mathbf{F}_m$ vector to satisfy the equation $\mathbf{F}_B = \mathbf{A}\mathbf{F}_m$ and $\mathbf{F}_B$ is as close as possible to $\mathbf{F}_B^d$.

We have:

$$\mathbf{F}_B = \mathbf{A}\mathbf{F}_m = \sum_{i=1}^{m} \mathbf{a}_i F_{m,i} \tag{4.13}$$

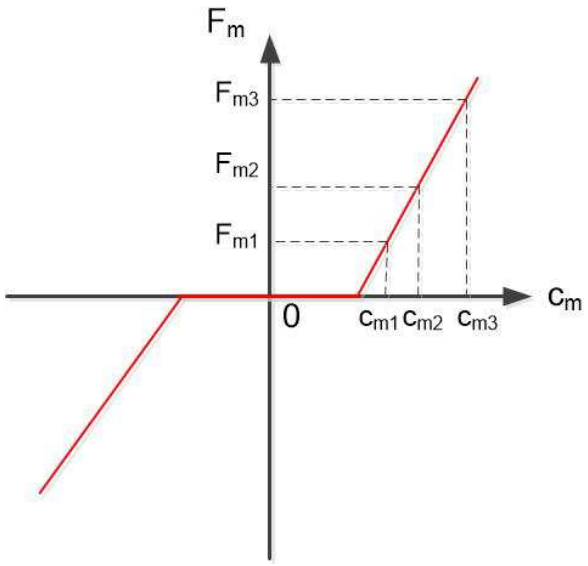


Figure 4.8 – Thruster characteristic approximation

where $\mathbf{a}_i$ is the i^{th} column of the matrix $\mathbf{A}$, and $F_{m,i}$ is the force magnitude of i^{th} thruster.

When one or more thrusters completely fail, the value of $F_{m,i} = 0$. Note that in the case that the i^{th} thruster is partly failed, the value of $F_{m,i}$ remains small (not addressed in this thesis). This is equivalent to consider a corresponding column $\mathbf{a}_i$ of the configuration matrix $\mathbf{A}$ equals to zero vector. Therefore, Equation (4.13) is equivalent as the equation:

$$\mathbf{F}_B = \mathbf{A}^{'}\mathbf{F}_m \tag{4.14}$$

where $\mathbf{A}^{'}$ matrix is the $\mathbf{A}$ matrix with one or more corresponding columns equal zero vectors.

We discuss hereafter two questions: conditions of the matrix $\mathbf{A}^{'}$ to guarantee the robustness, and what is the maximum number of thrusters failure?

For addressing these two questions, supposing that k-thrusters fail, and Equation (4.14) is a linear equation system with 6 equations (dimension of $\mathbf{F}_B$ is 6×1) and $(m - k)$ variables because the matrix $\mathbf{A}^{'}$ is $6 \times m$ with k columns are zero vectors. It is obvious to see that if $rank(\mathbf{A}^{'}) = 6$, for given $\mathbf{F}_B^d$, there always exits $\mathbf{F}_m$ such that $\mathbf{F}_B = \mathbf{A}^{'}\mathbf{F}_m$ and $\mathbf{F}_B$ is as close as possible to $\mathbf{F}_B^d$. This can be interpreted that $m - k \geq 6$ or $k \leq m - 6$. The condition of the configuration matrix and the maximum number of failure thrusters that guarantee the robustness of a marine robot are stated as:

1. *The maximum of faulty thrusters:* $m - 6$
2. *Robust condition: the rank of configuration matrix always equals to 6, i.e,* $rank(\mathbf{A}) = 6$, *if any columns, from one (1) to maximum* $(m - 6)$, *of* $\mathbf{A}$

matrix equal to zero vectors. If $rank(\mathbf{A}) < 6$, the system becomes underactuated, the guidance and control have to change to guarantee the robot's mission. This problem is not addressed in this thesis.

We define the robustness index as:

$$I_{ro} = rank(\mathbf{A}|_{\leq m-6}) = 6 \tag{4.15}$$

where $\mathbf{A}|_{\leq m-6}$ is the $\mathbf{A}$ matrix with the maximum number of columns being zero is $(m-6)$. This novel index will be verified in the solving process of the problem.

4.4 Configuration matrix design problem

With all performance indices discussed above, we design the problem hereafter:

$$\begin{aligned} &\min_{\mathbf{A}} \mathbf{V}(\mathbf{A}) = \min_{\mathbf{A}} [I_m, I_e, \frac{1}{I_w}, I_{re}]^T \\ &s.t \quad \mathbf{A} \in \mathbb{A} \end{aligned} \tag{4.16}$$

where $\mathbf{V}(\mathbf{A})$ is the objective function vector. $\mathbb{A}$ is the feasible set of the configuration matrix ($\mathbf{A}$) including constraints on configuration matrix ($\mathbf{A}$) and robust index. The inverse of the workspace index, $\frac{1}{I_w}$, is considered in Equation (4.16) because we want to maximize the workspace index.

It is obvious to see that a multiobjective optimization has unique solution when it has the convexity of each objective function in the objective vector and of the feasible set. However, our problem is non-convex and only one local optimal solution will be found. Note that this optimization problem is with respect to a matrix variable (*matrix optimization*), not to a vector variable. However, the optimization techniques for vector variables (*vector optimization*) can be applied here because we do not loose the physical meaning when converting a matrix variable to vector variable in this optimization problem (because of the independence of each column in the matrix derived from the independence of positions and orientations of thrusters).

Specifically, Equation (4.16) can be rewritten:

$$\begin{aligned} &\min_{\mathbf{A}} \mathbf{V}(\mathbf{A}) = \min_{\mathbf{A}} [I_m, I_e, \frac{1}{I_w}, I_{re}]^T \\ s.t \quad &\|\mathbf{u}_i\| = 1, i = 1, 2, ...m \\ &\|\tau_i\| \leq 1, i = 1, 2, ...m \\ &\tau_i^T \mathbf{u}_i = 0, i = 1, 2, ...m \\ &I_{ro} = rank(\mathbf{A}|_{\leq m-6}) = 6 \end{aligned} \tag{4.17}$$

The problem (4.17) is to minimize an objective vector $\mathbf{V}(\mathbf{A})$, including manipulability index, energetic index, inverse of workspace index, and reactive index, with respect to configuration matrix, $\mathbf{A}$, satisfies constraints of matrix structure itself and robust index. It is clear that this is a non-convex and multi-objective optimization problem which normally has many solutions. The methods for multiobjective optimization problem is presented in Chapter 3.

4.5 Searching for optimal solution

In this section, we find an optimal distribution (position and orientation) of thrusters of the marine robot. This means that one has to get $\mathbf{u}_i$ and $\mathbf{r}_i$ vectors for $i = 1, 2, ..., m$. These vectors can be extracted from configuration matrix $\mathbf{A}$ which is the solution of the problem (4.17). Recall that our problem (4.17) is the multi-objective optimization problem with non-convexity, and theoretically, this problem has many Pareto optimal solutions. The final choice depends on our choice. Our objective is to find one Pareto optimal solution for building the robot. Analyzing the underlying mathematical properties of the problem helps to simplify the solving process. Thus, the mathematical analysis of the problem is shown in the next section. In general, performance indices (manipulability, energetic, workspace, and reactive) will be considered successively and robustness index will be considered at the end.

4.5.1 Mathematical analysis

The configuration matrix $\mathbf{A}$ has the form as:

$$\mathbf{A} = \begin{pmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_m \\ \boldsymbol{\tau}_1 & \boldsymbol{\tau}_2 & \cdots & \boldsymbol{\tau}_m \end{pmatrix} \tag{4.18}$$

Let's set $\mathbf{B} = \mathbf{A}^T\mathbf{A}$. It is obvious to see that $\mathbf{B}$ is a $m \times m$ symmetric matrix where each element is denoted as b_{ij}. We have:

$$\begin{aligned} Tr(\mathbf{B}) &= \sum_{i=1}^{m} b_{ii} \\ &= \sum_{i=1}^{m} \lambda_i \end{aligned} \tag{4.19}$$

where λ_i is the i^{th} eigenvalue of matrix $\mathbf{B}$.

Hence,

$$\begin{aligned} \sum_{i=1}^{m} \lambda_i &= \sum_{i=1}^{m} \mathbf{u}_i^T \mathbf{u}_i + \boldsymbol{\tau}_i^T \boldsymbol{\tau}_i \\ &= \sum_{i=1}^{m} \|\mathbf{u}_i\|^2 + \|\boldsymbol{\tau}_i\|^2 \\ \sum_{i=1}^{m} \lambda_i &= \sum_{i=1}^{m} (1 + \|\boldsymbol{\tau}_i\|^2) \end{aligned} \tag{4.20}$$

4.5.1.1 Manipulability index

In the case of manipulability index optimization, the optimal configuration condition of matrix $\mathbf{A}$ is $cond(\mathbf{A}) = 1$. This means that the maximum singular value equals the minimum singular value, $\sigma_{max} = \sigma_{min}$. Note that the matrix $\mathbf{A}$ is the

$n \times m$ matrix with $n < m$ (since the system carries more actuators than DOFs). The matrix $\mathbf{A}$ has n non-zero singular values(we have to guarantee that $rank(\mathbf{A}) = n$), then the matrix $\mathbf{B}$ has n non-zero eigenvalues and $m - n$ zero eigenvalues. Note that the number of non-zero singular values (equivalent to the $rank(\mathbf{A})$) is also the number of actuable DoFs. If a system is fully-actuated, then $\mathbf{A}$ has to be full-rank.

Considering an optimal configuration w.r.t manipulability index, $cond(\mathbf{A}) = 1 \Rightarrow \sigma_{max} = \sigma_{min}$, we have $\lambda_i = \lambda_{max} = \lambda_{min} = \lambda_{opt-m}$ ($\sigma = \sqrt{\lambda_{opt-m}}$). The Equation (4.20) is rewritten:

$$\begin{aligned} n\lambda_{opt-m} &= m + \sum_{i=1}^{m} \|\boldsymbol{\tau}_i\|^2 \\ \lambda_{opt-m} &= \frac{m}{n} + \frac{1}{n}\sum_{i=1}^{m} \|\boldsymbol{\tau}_i\|^2 \end{aligned} \tag{4.21}$$

The fact that $\|\boldsymbol{\tau}_i\|^2 \leq 1$ (because of assumption $d_i = 1$ in $\mathbf{A}$ matrix), we have:

$$\lambda_{opt-m} \leq 2.\frac{m}{n} \tag{4.22}$$

Therefore, we have $\lambda_{opt-m}^{max} = 2\frac{m}{n}$ when $\|\boldsymbol{\tau}_i\|^2 = 1$.

4.5.1.2 Reactive index

In the singular value decomposition of a matrix, when $cond(\mathbf{A}) = 1$, the matrix $\mathbf{A}$ can be written as:

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T = \mathbf{U}[\sigma]_{n\times m}\mathbf{V}^T \tag{4.23}$$

where $\mathbf{U} \in \mathbb{R}^{n\times n}$, $\mathbf{V} \in \mathbb{R}^{m\times m}$ are orthogonal matrices, $\mathbf{S} = [\sigma]_{n\times m} = \begin{pmatrix} \sigma & 0 & \cdots & 0 \\ \vdots & \sigma & \cdots & 0 \\ 0 & \cdots & \sigma & 0 \end{pmatrix} \in \mathbb{R}^{n\times m}$.

The pseudo-inverse of matrix $\mathbf{A}$ is $\mathbf{A}^+$ can be written:

$$\mathbf{A}^+ = \mathbf{V}\mathbf{S}^+\mathbf{U}^T \tag{4.24}$$

where $\mathbf{S}^+ = [\frac{1}{\sigma}]_{m\times n} = \begin{pmatrix} \frac{1}{\sigma} & \cdots & 0 \\ \vdots & \frac{1}{\sigma} & 0 \\ 0 & 0 & \frac{1}{\sigma} \\ 0 & \cdots & 0 \end{pmatrix} \in \mathbb{R}^{m\times n}$.

Our objective with reactive index is to minimize the $\|\mathbf{A}^+\|$. From Equation (4.24), the reactive index $I_{re} = \|\mathbf{A}^+\| = \frac{1}{\sigma}$, the minimum value of reactive index is equivalent with the maximum value of σ. This leads to the equality of Equation (4.22) holds.

In order to minimize the reactive index and satisfy manipulability index, the configuration matrix $\mathbf{A}$ has the structure:

$$\begin{aligned}\mathbf{A} &= \mathbf{U}\mathbf{S}\mathbf{V}^T \\ &= \mathbf{U}\begin{pmatrix}\sigma & 0 & \cdots & 0 & 0 & 0\\ 0 & \sigma & 0 & \cdots & 0 & 0\\ 0 & 0 & \sigma & 0 & \cdots & 0\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & 0 & \sigma & 0 & 0\end{pmatrix}\mathbf{V}^T\end{aligned} \tag{4.25}$$

where $\mathbf{S}(n \times m)$ is like-diagonal and $\sigma = \sigma^{max} = \sqrt{\lambda_{opt-m}^{max}} = \sqrt{2\frac{m}{n}}$; $\mathbf{U}(n \times n)$ and $\mathbf{V}(m \times m)$ are orthogonal matrices ($\mathbf{U}\mathbf{U}^T = \mathbf{I}, \mathbf{V}\mathbf{V}^T = \mathbf{I}$). This results can be used as initial value of numerical optimization process and useful for solving the problem.

4.5.1.3 Energetic index

First, we introduce a proposition as follows:

Proposition 4.5.1 *: Let* $\mathbf{M}$ *be a* $p \times q$ *matrix* ($p \geq q$), $\mathbf{M} \in \mathbb{R}^{p\times q}$. *For all* $\mathbf{x} \in \mathbb{R}^q$, *if* $\mathbf{M} = \mathbf{P}\mathbf{\Sigma}\mathbf{Q}^T$, *where* $\mathbf{P} \in \mathbb{R}^{p\times p}, \mathbf{Q} \in \mathbb{R}^{q\times q}$ *are orthogonal matrices,* $\mathbf{\Sigma} = \begin{pmatrix}\mu & 0 & \cdots & 0\\ 0 & \mu & \cdots & 0\\ 0 & \cdots & \mu & 0\\ 0 & \cdots & 0 & \mu\\ \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & 0 & 0\end{pmatrix} \in \mathbb{R}^{p\times q}$ *then* $\|\mathbf{M}\mathbf{x}\| = \|\mathbf{M}\|\|\mathbf{x}\|$.

Proof : We have:

$$\|\mathbf{M}\mathbf{x}\|^2 = (\mathbf{M}\mathbf{x})^T(\mathbf{M}\mathbf{x}) = \mathbf{x}^T\mathbf{M}^T\mathbf{M}\mathbf{x} \tag{4.26}$$

with $\mathbf{M} = \mathbf{P}\mathbf{\Sigma}\mathbf{Q}^T$

$$\begin{aligned}\|\mathbf{M}\mathbf{x}\|^2 &= \mathbf{x}^T(\mathbf{P}\mathbf{\Sigma}\mathbf{Q}^T)^T(\mathbf{P}\mathbf{\Sigma}\mathbf{Q}^T)\mathbf{x}\\ &= \mathbf{x}^T\mathbf{Q}\mathbf{\Sigma}^T\mathbf{P}^T\mathbf{P}\mathbf{\Sigma}\mathbf{Q}^T\mathbf{x}\\ &= \mathbf{x}^T\mathbf{Q}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{Q}^T\mathbf{x}\end{aligned} \tag{4.27}$$

We have:

$$\mathbf{\Sigma}^T\mathbf{\Sigma} = \begin{pmatrix} \mu & 0 & \cdots & 0 \\ 0 & \mu & \cdots & 0 \\ 0 & \cdots & \mu & 0 \\ 0 & \cdots & 0 & \mu \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \end{pmatrix}^T \begin{pmatrix} \mu & 0 & \cdots & 0 \\ 0 & \mu & \cdots & 0 \\ 0 & \cdots & \mu & 0 \\ 0 & \cdots & 0 & \mu \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \end{pmatrix}$$
$$= \begin{pmatrix} \mu^2 & 0 & \cdots & 0 \\ 0 & \mu^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & \mu^2 \end{pmatrix} = \mu^2\mathbf{I} \tag{4.28}$$

where $\mathbf{I}$ is $q \times q$ identity matrix.

Replacing Equation (4.28) to (4.27), we have:

$$\begin{aligned} \|\mathbf{Mx}\|^2 &= \mathbf{x}^T\mathbf{V}\mu^2\mathbf{I}\mathbf{V}^T\mathbf{x} \\ &= \mu^2\mathbf{x}^T\mathbf{x} = \|\mathbf{M}\|^2\|\mathbf{x}\|^2 \end{aligned} \tag{4.29}$$

Therefore, $\|\mathbf{Mx}\| = \|\mathbf{M}\|\|\mathbf{x}\|$. ∎

The energetic index is stated as:

$$I_e = \frac{1}{S}\int_S (w_{ef}\|\mathbf{A}^+(\mathbf{F}_B^d(\mathbf{f})\| + w_{e\Gamma}\|\mathbf{A}^+\mathbf{F}_B^d(\mathbf{\Gamma})\|)dS \tag{4.30}$$

Choose $w_{ef} = w_{e\Gamma} = 1$ (because desired force vectors, $\mathbf{F}_B^d(\mathbf{f}), \mathbf{F}_B^d(\mathbf{\Gamma})$, are unit), we have:

$$I_e = \frac{1}{S}\int_S (\|\mathbf{A}^+\mathbf{F}_B^d(\mathbf{f})\| + \|\mathbf{A}^+\mathbf{F}_B^d(\mathbf{\Gamma})\|)dS \tag{4.31}$$

In case the minimum of reactive index and manipulability index, the configuration matrix $\mathbf{A}(n \times m)$ has the form as Equation (4.25), therefore the pseudo-inverse matrix $\mathbf{A}^+(m \times n, \quad m > n)$ has the structure as:

$$\mathbf{A}^+ = \mathbf{V}\mathbf{S}^+\mathbf{U}^T = \mathbf{V}\begin{pmatrix} \frac{1}{\sigma} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma} & \cdots & 0 \\ 0 & \cdots & \frac{1}{\sigma} & 0 \\ 0 & \cdots & 0 & \frac{1}{\sigma} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \end{pmatrix}\mathbf{U}^T \tag{4.32}$$

where $\mathbf{V}, \mathbf{U}$ are orthogonal matrices.

It is clear that matrix $\mathbf{A}^+$ satisfies the condition of proposition 4.5.1. Applying this proposition, we have: $\|\mathbf{A}^+\mathbf{F}_B^d(\mathbf{f})\| = \|\mathbf{A}^+\|\|\mathbf{F}_B^d(\mathbf{f})\|$ and $\|\mathbf{A}^+\mathbf{F}_B^d(\mathbf{\Gamma})\| =$

$\|\mathbf{A}^+\|\|\mathbf{F}_B^d(\mathbf{\Gamma})\|$. Therefore, Equation (4.31) becomes:

$$\begin{aligned} I_e &= \frac{1}{S}\int_S (\|\mathbf{A}^+\|\|\mathbf{F}_B^d(\mathbf{f})\| + \|\mathbf{A}^+\|\|\mathbf{F}_B^d(\mathbf{\Gamma})\|)dS \\ &= \frac{1}{S}\|\mathbf{A}^+\|\int_S (\|\mathbf{F}_B^d(\mathbf{f})\| + \|\mathbf{F}_B^d(\mathbf{\Gamma})\|)dS \\ &= 2\|\mathbf{A}^+\| \end{aligned} \tag{4.33}$$

For aforementioned mathematical analysis of the energetic index, we can see that the energetic index belongs to the norm of pseudo-inverse of configuration matrix, $I_{re} = 2\|\mathbf{A}^+\|$, when the configuration matrix $\mathbf{A}$ has the form of (4.25).

4.5.1.4 Workspace index

We discuss about the upper-bound of workspace index. For units consistency, the workspace index for force space and for torque space are investigate separately, denoted as I_{wf} and $I_{w\tau}$ respectively. Recall that the objective of workspace index is to maximize the volume of resulting force space ($\mathbf{F}_B$ space) including resulting space for force and resulting space for torque with given the thrusters force space ($\mathbf{F}_m$ space).

The fact that for all vector $\mathbf{F}_m \in \mathbb{R}^m$, $\|\mathbf{A}\mathbf{F}_m\| \leq \|\mathbf{A}\|\|\mathbf{F}_m\|$. The volume of the resulting force space is maximum when the equality holds. Following Fig 4.9,

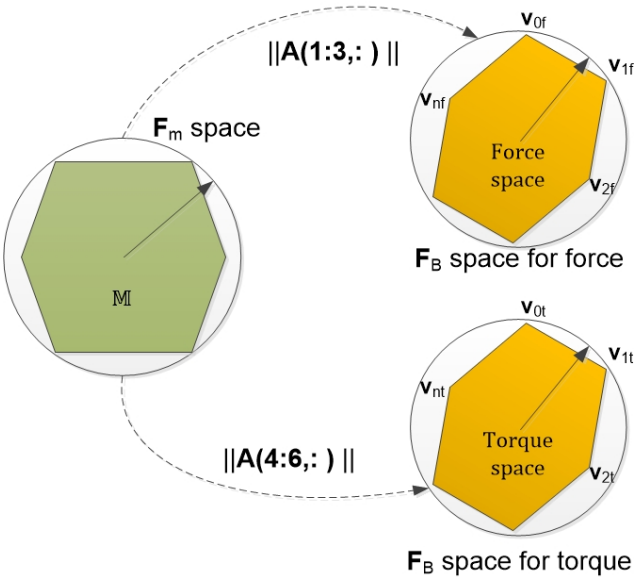


Figure 4.9 – Upper-bound of resulting force space

the volume of resulting force spaces ($\mathbf{F}_B$)(force and torque spaces) are always less than the volume of exterior hyper-sphere of $\mathbf{F}_B$ spaces of force and torque (the

circumscribed spheres or maybe not). This means that:

$$\begin{aligned} I_{wF} &\leq Volume(\mathbf{B}(R1)) \\ I_{wT} &\leq Volume(\mathbf{B}(R2)) \end{aligned} \tag{4.34}$$

where $\mathbf{B}(R1)$ and $\mathbf{B}(R2)$ are an Euclidean balls of radius $R1 = \|\mathbf{A}(\mathbf{1:3},:)\|\|\mathbf{F}_m\| = \|\mathbf{A}_1\|\|\mathbf{F}_m\|$ and $R2 = \|\mathbf{A}(\mathbf{4:6},:)\|\|\mathbf{F}_m\| = \|\mathbf{A}_2\|\|\mathbf{F}_m\|$ respectively; $\mathbf{A}(\mathbf{1:3},:)$ is the $\mathbf{A}$ matrix with three first rows, and $\mathbf{A}(\mathbf{4:6},:)$ is the $\mathbf{A}$ matrix with three last rows.

The fact that n-dimensional volume of an Euclidean ball of radius R in n-dimensional Euclidean space is:

$$V_n(R) = \begin{cases} \frac{\pi^k}{k!}R^{2k}, & \text{if } n = 2k \\ \frac{2^{k+1}\pi^k}{(2k+1)!!}R^{2k+1}, & \text{if } n = 2k+1. \end{cases} \tag{4.35}$$

where $(2k+1)!! = 1.3.5...(2k-1).(2k+1)$.

Proposition 4.5.2 *: If the configuration matrix* $\mathbf{A}$ *has the form of* (4.25) *then* $cond(\mathbf{A}_1) = cond(\mathbf{A}_2) = 1$ *and* $\|\mathbf{A}_1\| = \|\mathbf{A}_2\| = \sigma$.

Proof :

We have:

$$\begin{aligned} \mathbf{A}\mathbf{A}^T &= (\mathbf{U}\mathbf{S}\mathbf{V}^T)(\mathbf{U}\mathbf{S}\mathbf{V}^T)^T = \mathbf{U}\mathbf{S}\mathbf{V}^T\mathbf{V}\mathbf{S}^T\mathbf{U}^T \\ &= \mathbf{U}\mathbf{S}\mathbf{S}^T\mathbf{U}^T = \sigma^2\mathbf{I} \end{aligned} \tag{4.36}$$

On the other hand:

$$\begin{aligned} \mathbf{A}\mathbf{A}^T &= \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{pmatrix}\begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{pmatrix}^T = \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{pmatrix}(\mathbf{A}_1^T\mathbf{A}_2^T) \\ &= \begin{pmatrix} \mathbf{A}_1\mathbf{A}_1^T & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2\mathbf{A}_2^T \end{pmatrix} \end{aligned} \tag{4.37}$$

From (4.36) and (4.37), we have:

$$\begin{aligned} \mathbf{A}_1\mathbf{A}_1^T &= \sigma^2\mathbf{I}_1 \\ \mathbf{A}_2\mathbf{A}_2^T &= \sigma^2\mathbf{I}_2 \end{aligned} \tag{4.38}$$

where $\mathbf{I}_1$ and $\mathbf{I}_2$ are partitioned matrices of matrix $\mathbf{I}$.

From (4.38) and the uniqueness of singular value decomposition [Trefethen 1997], it is obvious to get the structures of $\mathbf{A}_1$ and $\mathbf{A}_2$ are the same as (4.25) with different dimensions. Therefore, $cond(\mathbf{A}_1) = cond(\mathbf{A}_1) = 1$ and $\|\mathbf{A}_1\| = \|\mathbf{A}_2\| = \sigma$. ∎

From (4.34) and (4.35) and proposition 4.5.2, it is obvious to get the upper-bound of resulting spaces of force and torque of the system, and then the upper-bound of workspace index. Normally, the weighting coefficients in workspace index are chosen as 1 because of our assumption of d_i.

For robustness index, we have to make sure that configuration matrix, $\mathbf{A}$, has to be full-rank. Therefore, this will be checked at the final step.

4.5.2 The optimization process

Based on the above mathematical analyses, goal attainment method is chosen to solve the problem (4.17) with given desired values. The idea of this method is to minimize the deviation of desired values and getting values. One advantage of goal attainment method is that the problem do not need to normalize to dimensionless problem. The solution of this method is proven to be Pareto optimal [Gembicki 1975]. This method is also suitable when the feasible objective set is non-convex. All Pareto optimal solutions can be found by changing the weighting vector.

Our problem using goal attainment approach becomes:

$$\begin{aligned} &\min_{\mathbf{A},\gamma} \gamma \\ &s.t \quad \mathbf{A} \in \bar{\mathbb{A}} \\ &\qquad \mathbf{V}(\mathbf{A}) - \mathbf{w}\gamma \leq \mathbf{V}_{goal} \end{aligned} \tag{4.39}$$

where $\bar{\mathbb{A}} = \mathbb{A} \setminus I_{ro}$, i.e, $\mathbf{A}$ set without robust index I_{ro}, γ is a slack vector variable, $\mathbf{V}_{goal} = [I_m^d \quad I_e^d \quad \frac{1}{I_w^d} \quad I_{re}^d]$ is the desired objective vector, $\mathbf{w}$ is a attainment vector which can be chosen by ourselves. The goal attainment method with two objective functions is illustrated in Fig 4.10. By altering attainment vector, $\mathbf{w}$, we get Pareto optimal solutions. The chosen solution is about choosing this attainment vector.

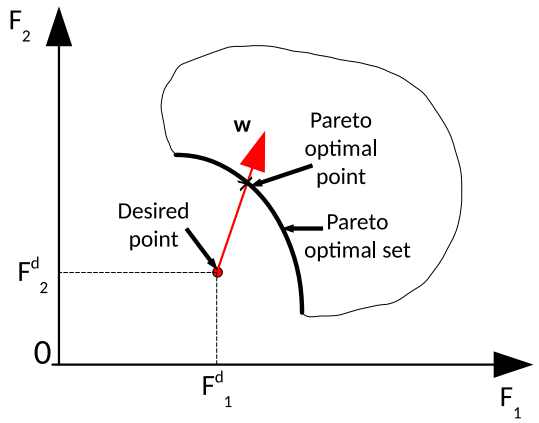


Figure 4.10 – Goal attainment method with two objective functions

Therefore, the problem solving process includes two phases:

1. Phase 1: Find one Pareto solution (see Definition 7 in chapter 3) of configuration matrix with goal attainment method. Note that finding the set of all Pareto solutions, called Pareto front, is not considered in the thesis.
2. Phase 2: Check robustness index of the chosen solution in phase 1.

4.6 Conclusion

A proper configuration design of a robot can optimize its operation ability, save energy, and reduce building cost as well. A static configuration design problem was proposed and described in this section with respect to performance indices including manipulability, energetic, workspace, reactive, and robustness criteria. Specifically, manipulability index represents isotropy ability of robot, energetic index is considered as the energy consumption when an unit desired control vector changes all over hyper-sphere, workspace index represents attainable spaces of desired control vectors (force and torque spaces), reactive index expresses how fast that the robot can change the direction of resulting force vector, and finally robustness index represents the independently acting ability of along 6 DoFs. All indices were defined and exploited on properties of configuration matrix, $\mathbf{A}$. This is a multi-objective optimization problem in which objective functions are non-convex. By mathematical analysis, goal attainment method was suitable and chosen to solve it because we can know desired or lower-upper bound solutions of each objective function. A solving procedure including two phases was designed. In the phase 1, goal attainment method is applied to find one Pareto solution with respect to manipulability, energetic, workspace, and reactive indices. In phase 2, robustness index will be check with the optimal solution. If it satisfies robustness index, the procedure stops, if not, return to phase 1 to find another Pareto solution. Note that just one Pareto solution will be found and chosen for simulation and experiments in the next chapter. Finding a set of all Pareto solutions, Pareto front, will be a future direction.

Chapter 5

Static Configuration: Simulations and Experiments

Contents

Simulations and experimental results of static configuration design problem are presented. Note that the result is just one of Pareto solutions (see Definition 7) and Pareto front is not considered.

5.1 Simulations

We have designed a marine robot with $m = 8$ thrusters and $n = 6$ degrees of freedom. Two cases are simulated:

1. General case: we have to identify both the positions and orientations of 8 thrusters optimizing the performance indices.
2. Given-position case: the thrusters are installed at the corners of a cube, we only have to determine the directions of thrusters.

In our solving and simulation, thrusters characteristics are chosen as in [Ropars 2018], in which the maximum and minimum values of thrusters forces are as $F_{imax} = 1.1N$ and $F_{imin} = -0.4N$ respectively. The desired values of performance indices are subsequently $I_m = 1, I_e = 1.2248, I_{wF} = 597.7, I_{wT} = 597.7, I_{re} = 0.6124$ ($\sigma^{max} = \sqrt{2\frac{m}{n}} = 1.6330$). See Table 5.1 for more details.

Index	Optimal formula and condition	Desired Value
I_m	$\sigma_{max} = \sigma_{min}$	1
I_e	$2 \left\|\mathbf{A}^+\right\|$	1.2248
$\frac{1}{I_w}$	see Equation 4.34 and 4.35 and $\frac{1}{I_w} = \frac{1}{I_{wF}} + \frac{1}{I_{wT}}$	0.0033
I_{re}	$\frac{1}{\sigma^{max}}$	0.6124

Table 5.1 – Desired values of indices

Configuration matrix	Optimal value
$\mathbf{A} = \begin{pmatrix} -0.8891 & -0.3645 & 0.5438 & 0.9879 & 0.3134 & 0.0148 & 0.0495 & 0.6090 \\ -0.0985 & -0.3036 & -0.5911 & -0.0608 & -0.9493 & 0.0515 & 0.8919 & 0.7158 \\ 0.4471 & 0.8803 & 0.5957 & 0.1429 & 0.0260 & 0.9986 & 0.4495 & 0.3417 \\ -0.4308 & 0.4701 & -0.8386 & 0.0379 & -0.1336 & 0.5628 & -0.9972 & 0.4758 \\ 0.5107 & 0.7561 & -0.4103 & 0.9868 & -0.0712 & -0.8259 & 0.0690 & 0.0149 \\ -0.7441 & 0.4554 & 0.3583 & 0.1577 & -0.9885 & 0.0342 & -0.0272 & -0.8794 \end{pmatrix}$	$Fval = \begin{pmatrix} 1.0000 \\ 1.2200 \\ 0.0050 \\ 0.6124 \end{pmatrix}$

Table 5.2 – Configuration matrix in general case by solving the problem (4.39)

5.1.1 General case

In this case the positions and orientations of thrusters are not known. The problem (4.39) is solved as follows. Because of assumption $d_i = 1$ (in Equation 4.2), all thrusters will be placed in a sphere and it is called Ball robot, that is virtual robot like a ball and directions and positions of thrusters are determined to achieve performance indices. The same kind of the robot, called *SamoS,* in literature can be seen in [Pierrot 1998], but it is not redundant.

5.1.1.1 Phase 1

Optimization Matlab toolbox is used to solve the problem (4.39) with $\mathbf{V}_{goal} = [I_m^d \quad I_e^d \quad \frac{1}{I_w^d} \quad I_{re}^d] = [1 \quad 1.2248 \quad 0.0036 \quad 0.6124]$, the constraint set $\bar{\mathbb{A}} = \{\mathbf{A} \in \mathbb{R}^{6\times 8} / \|\mathbf{u}_i\| = 1, \|\boldsymbol{\tau}_i\| \leq 1, \boldsymbol{\tau}_i^T \mathbf{u_i} = 0\}$, the attainment vector $\mathbf{w} = [0 \quad 0 \quad 0 \quad 0.0036]^T$.

The simulation results are shown in Figs 5.1, 5.2(a), and 5.2(b). The configuration matrix $\mathbf{A}$ and optimal values are shown in Table 5.2. The robot's shape is shown in Fig 5.1, called Ball robot. Position and direction vectors of thrusters are presented in Table 5.3 (note that directions of thruster are the three first rows of configuration matrix, and positions of thrusters can be interpolated from three last rows of the matrix, see Proposition A.1.2 in Appendix A). Furthermore, we can see that the isotropy property of robot is guaranteed by attainable force and torque spaces (see Figures 5.2(a), 5.2(b)). From Table 5.2, the actual values of manipulability index, energetic index, and reactive index are almost the same desired values. However, the actual value of workspace index is under-attainment of desired value with an attainment factor.

5.1.1.2 Phase 2

In this phase, the robustness index is checked. The optimal configuration matrix $\mathbf{A}$ in Table 5.2 satisfies the robust constraint. It means that if one or two thrusters

Thruster No.	Position	Direction	Direction in angles(degree)
1	$\begin{pmatrix} x = -0.1558 \\ y = -0.8542 \\ z = -0.4961 \end{pmatrix}$	$\begin{pmatrix} x = -0.8891 \\ y = -0.0985 \\ z = 0.4471 \end{pmatrix}$	$\begin{pmatrix} azimuth = -173.6782 \\ elevation = 26.5564 \end{pmatrix}$
2	$\begin{pmatrix} x = -0.8043 \\ y = 0.5794 \\ z = -0.1316 \end{pmatrix}$	$\begin{pmatrix} x = -0.3645 \\ y = -0.3036 \\ z = 0.8803 \end{pmatrix}$	$\begin{pmatrix} azimuth = -140.2083 \\ elevation = 61.6806 \end{pmatrix}$
3	$\begin{pmatrix} x = 0.0326 \\ y = -0.6944 \\ z = -0.7188 \end{pmatrix}$	$\begin{pmatrix} x = 0.5438 \\ y = -0.5911 \\ z = 0.5957 \end{pmatrix}$	$\begin{pmatrix} azimuth = -47.3866 \\ elevation = 36.5631 \end{pmatrix}$
4	$\begin{pmatrix} x = -0.1506 \\ y = -0.1504 \\ z = 0.9771 \end{pmatrix}$	$\begin{pmatrix} x = 0.9879 \\ y = -0.0608 \\ z = 0.1429 \end{pmatrix}$	$\begin{pmatrix} azimuth = -3.5218 \\ elevation = 8.2154 \end{pmatrix}$
5	$\begin{pmatrix} x = 0.9402 \\ y = 0.3064 \\ z = -0.1491 \end{pmatrix}$	$\begin{pmatrix} x = 0.3134 \\ y = -0.9493 \\ z = 0.0260 \end{pmatrix}$	$\begin{pmatrix} azimuth = -71.7300 \\ elevation = 1.4898 \end{pmatrix}$
6	$\begin{pmatrix} x = 0.8265 \\ y = 0.5615 \\ z = -0.0412 \end{pmatrix}$	$\begin{pmatrix} x = 0.0148 \\ y = 0.0515 \\ z = 0.9986 \end{pmatrix}$	$\begin{pmatrix} azimuth = 73.9665 \\ elevation = 86.9285 \end{pmatrix}$
7	$\begin{pmatrix} x = -0.0552 \\ y = -0.4459 \\ z = 0.8934 \end{pmatrix}$	$\begin{pmatrix} x = 0.0495 \\ y = 0.8919 \\ z = 0.4495 \end{pmatrix}$	$\begin{pmatrix} azimuth = 86.8234 \\ elevation = 26.7118 \end{pmatrix}$
8	$\begin{pmatrix} x = -0.6346 \\ y = 0.6981 \\ z = -0.3315 \end{pmatrix}$	$\begin{pmatrix} x = 0.6090 \\ y = 0.7158 \\ z = 0.3417 \end{pmatrix}$	$\begin{pmatrix} azimuth = 49.6090 \\ elevation = 19.9804 \end{pmatrix}$

Table 5.3 – Positions and orientations of 8 thrusters in general case (one Pareto solution)

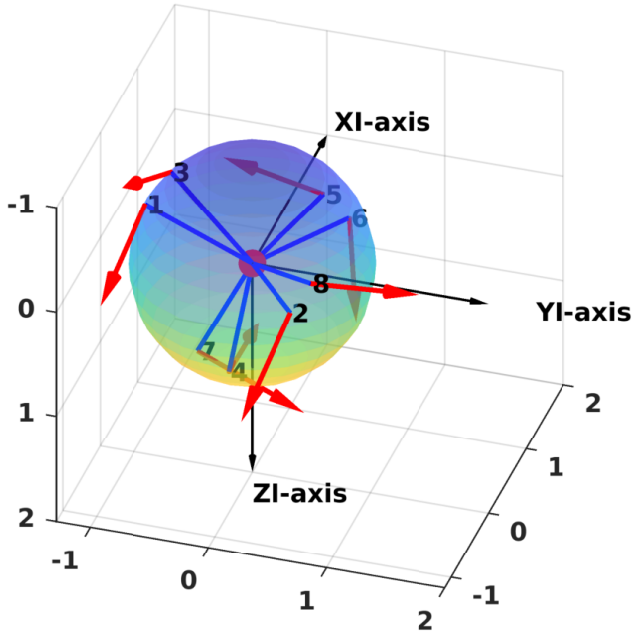
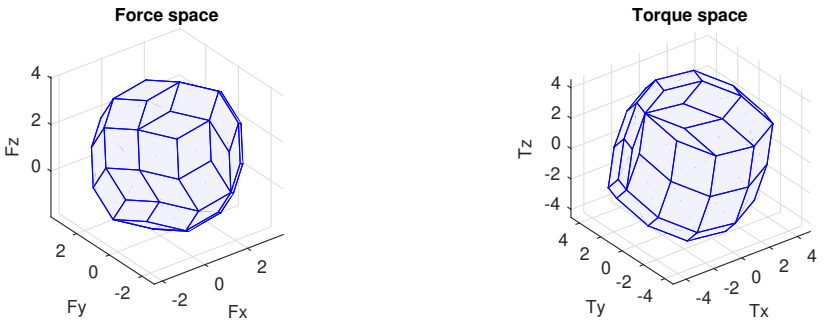


Figure 5.1 – Robot design (general case-unknown positions and directions of thrusters) (Ball robot)



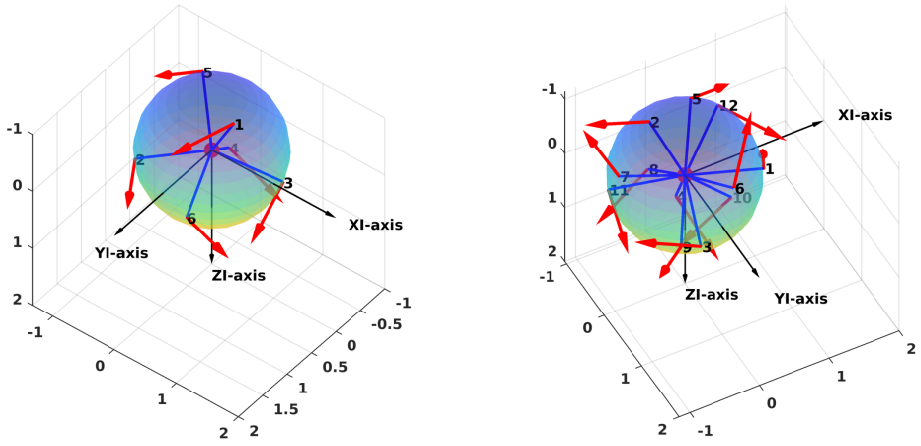
(a) Attainable force space

(b) Attainable torque space

Figure 5.2 – Attainable force and torque spaces in general case (unknown positions and directions of thrusters)

fail, the robot is still able to act along the 6 DoFs independently.

Furthermore, robustness index can be evaluated with two other different configurations, ball robot with 6 thrusters and with 12 thrusters (see Figure 5.3). The robustness index satisfies in case of 12 thrusters and does not achieve in case of 6 thrusters.



(a) Ball robot with 6 thrusters (b) Ball robot with 12 thrusters

Figure 5.3 – Ball robot with 6 and 12 thrusters in general case

Configuration matrix									Optimal value
$\mathbf{A} =$	0.0836	0.6616	−0.8122	0.4785	−0.6616	−0.0836	−0.4785	−0.8122	$Fval = \begin{pmatrix} 1.0000 \\ 1.2200 \\ 0.0050 \\ 0.6124 \end{pmatrix}$
	0.7452	0.7452	0.3337	0.3337	0.7452	0.7452	0.3337	−0.3337	
	0.6616	−0.0836	−0.4785	−0.8122	0.0836	−0.6616	0.8122	−0.4785	
	−0.8122	0.4785	−0.0836	−0.6616	−0.4785	0.8122	0.6616	−0.0836	
	−0.3337	−0.3337	0.7452	0.7452	−0.3337	−0.3337	0.7452	−0.7452	
	0.4785	0.8122	0.6616	−0.0836	−0.8122	−0.4785	0.0836	0.6616	

Table 5.4 – Configuration matrix in given position case

5.1.2 Given position case

In this case the positions of thrusters are given, at corners of a cube. We just only have to find their orientations. The number of variables in the problem (4.39) is reduced. The desired vector and attainment vector are the same as in general case.

5.1.2.1 Phase 1

Similarly, Matlab optimization toolbox is used to solve our problem and simulation results are shown in Figs 5.4, 5.5(a), 5.5(b), and Table 5.4. The robot's shape and directions of thrusters are depicted as red arrows in Fig 5.4, and position and direction vectors of thrusters are displayed in Table 5.5. Similar to the general case, the isotropy property is also achieved in this case by looking at attainable force and torque spaces (Figs 5.5(a) and 5.5(b)). We can see that the getting objective values in Table 5.4 are reached at desired values except workspace index.

Thruster No.	Position	Direction	Direction in angles(degree)
1	$\begin{pmatrix} x = 0.5773 \\ y = -0.5773 \\ z = 0.5773 \end{pmatrix}$	$\begin{pmatrix} x = 0.0836 \\ y = 0.7452 \\ z = 0.6616 \end{pmatrix}$	$\begin{pmatrix} azimuth = 83.5991 \\ elevation = 41.4213 \end{pmatrix}$
2	$\begin{pmatrix} x = 0.5773 \\ y = -0.5773 \\ z = -0.5773 \end{pmatrix}$	$\begin{pmatrix} x = 0.6616 \\ y = 0.7452 \\ z = -0.0836 \end{pmatrix}$	$\begin{pmatrix} azimuth = 48.4008 \\ elevation = -4.7955 \end{pmatrix}$
3	$\begin{pmatrix} x = 0.5773 \\ y = 0.5773 \\ z = -0.5773 \end{pmatrix}$	$\begin{pmatrix} x = -0.8122 \\ y = 0.3337 \\ z = -0.4785 \end{pmatrix}$	$\begin{pmatrix} azimuth = 157.6642 \\ elevation = -28.5877 \end{pmatrix}$
4	$\begin{pmatrix} x = 0.5773 \\ y = 0.5773 \\ z = 0.5773 \end{pmatrix}$	$\begin{pmatrix} x = 0.4785 \\ y = 0.3337 \\ z = -0.8122 \end{pmatrix}$	$\begin{pmatrix} azimuth = 34.8914 \\ elevation = -54.3120 \end{pmatrix}$
5	$\begin{pmatrix} x = -0.5773 \\ y = -0.5773 \\ z = 0.5773 \end{pmatrix}$	$\begin{pmatrix} x = -0.6616 \\ y = 0.7452 \\ z = 0.0836 \end{pmatrix}$	$\begin{pmatrix} azimuth = 131.5992 \\ elevation = 4.7955 \end{pmatrix}$
6	$\begin{pmatrix} x = -0.5773 \\ y = -0.5773 \\ z = -0.5773 \end{pmatrix}$	$\begin{pmatrix} x = -0.0836 \\ y = 0.7452 \\ z = -0.6616 \end{pmatrix}$	$\begin{pmatrix} azimuth = 96.4009 \\ elevation = -41.4213 \end{pmatrix}$
7	$\begin{pmatrix} x = -0.5773 \\ y = 0.5773 \\ z = -0.5773 \end{pmatrix}$	$\begin{pmatrix} x = -0.4785 \\ y = 0.3337 \\ z = 0.8182 \end{pmatrix}$	$\begin{pmatrix} azimuth = 145.1086 \\ elevation = 54.3120 \end{pmatrix}$
8	$\begin{pmatrix} x = -0.5773 \\ y = 0.5773 \\ z = 0.5773 \end{pmatrix}$	$\begin{pmatrix} x = -0.8122 \\ y = -0.3337 \\ z = -0.4785 \end{pmatrix}$	$\begin{pmatrix} azimuth = -157.6642 \\ elevation = -28.5877 \end{pmatrix}$

Table 5.5 – Positions and orientations of 8 thrusters in given-position case (one Pareto solution)

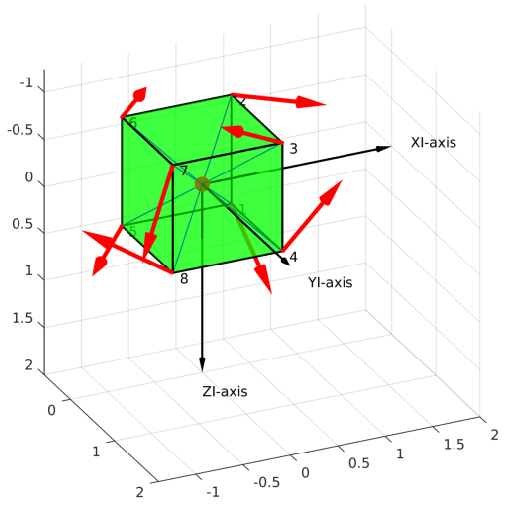
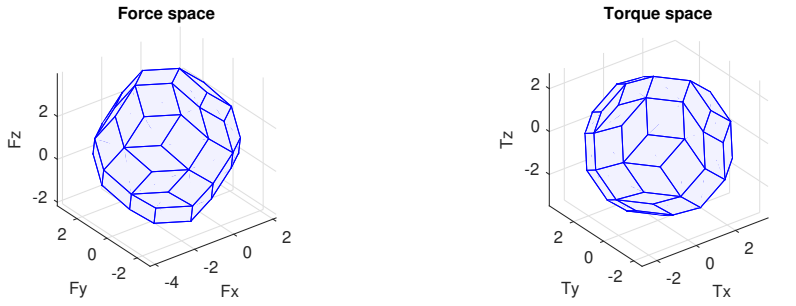


Figure 5.4 – Robot design (given position case) in which 8 thrusters are installed at vertices of cube shape and directions of thrusters are along red arrow lines



(a) Attainable force space

(b) Attainable torque space

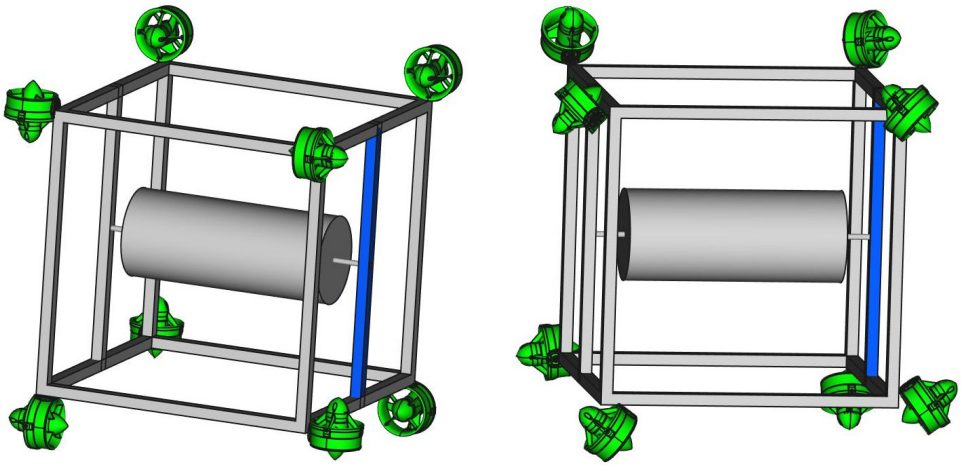
Figure 5.5 – Attainable torque space (given position case - Cube shape)

5.1.2.2 Phase 2

The optimal configuration matrix $\mathbf{A}$ in Table 5.4 satisfies the conditions of robustness index.

5.1.3 A comparison of the two configurations of Cube robot (given position)

In this section, a comparison of two configurations is illustrated. The first one is an arbitrary configuration (denoted as $\mathbf{C}^1$) in which the thrusters are distributed vertically or horizontally. In practice, this configuration is easier to install and displayed at Figure 5.6(a) (3D model). The configuration matrix of $\mathbf{C}^1$, denoted



(a) 3D model of Cube robot in $\mathbf{C^1}$ configuration (b) 3D model of Cube robot in $\mathbf{C^2}$ configuration

Figure 5.6 – 3D model of Cube robot in two configurations $\mathbf{C^1}$ and $\mathbf{C^2}$

$\mathbf{A}_1$, is shown in Equation (5.1).

$$\mathbf{A}_1 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & -1 \\ 0.27 & 0 & -0.27 & 0.27 & 0.27 & 0.27 & 0 & 0.27 \\ 0 & -0.27 & 0.27 & 0 & 0 & 0.27 & -0.27 & -0.27 \\ 0.27 & -0.27 & 0 & 0.27 & 0.27 & 0 & 0.27 & 0 \end{pmatrix} \tag{5.1}$$

The second one (denoted as $\mathbf{C}^2$) is an optimal configuration, denoted as $\mathbf{A}_2$, which is the solution of our optimization problem in given position case, and the optimal configuration matrix is shown in Equation (5.2).

$$\mathbf{A}_2 = \begin{pmatrix} 0.6616 & -0.8122 & 0.4785 & 0.0836 & -0.0836 & -0.4785 & -0.8122 & -0.6616 \\ 0.7452 & 0.3337 & 0.3337 & 0.7452 & 0.7452 & 0.3337 & -0.3337 & 0.7452 \\ -0.0836 & -0.4785 & -0.8122 & 0.6616 & -0.6616 & 0.8122 & -0.4785 & 0.0836 \\ 0.1608 & 0.0111 & -0.2459 & -0.3708 & 0.3642 & 0.2015 & 0.0011 & -0.1658 \\ -0.0989 & 0.3556 & 0.3633 & -0.0989 & -0.1056 & 0.3508 & -0.3456 & -0.1056 \\ 0.3906 & 0.2292 & 0.0044 & 0.1583 & -0.1649 & -0.0254 & 0.2392 & -0.3708 \end{pmatrix} \tag{5.2}$$

Note that the configuration matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ are calibrated with corresponding

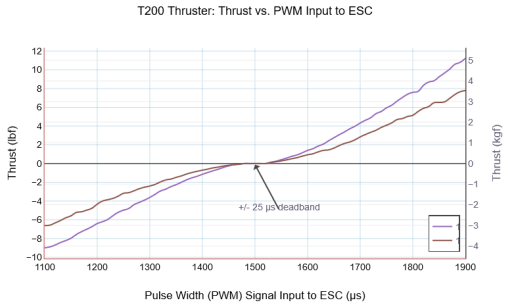
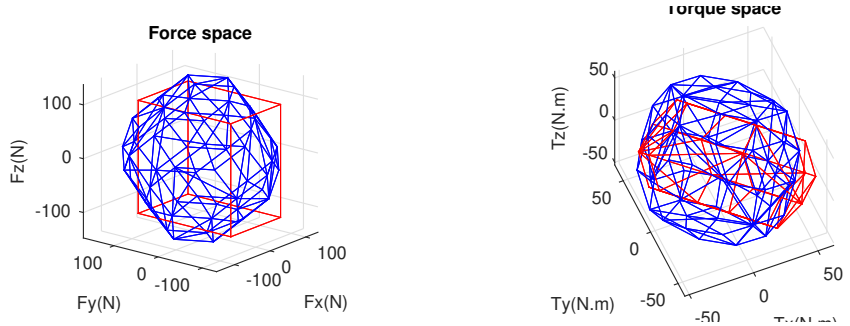


Figure 5.7 – Thruster characteristic(BlueRobotics) [BlueRobotics]

geometrical properties of real cube robot built in Polytech and LIRMM Institute,

Montpellier University. The attainable force and torque spaces corresponding with two configurations $\mathbf{C}^1$ and $\mathbf{C}^2$ are illustrated in Figures 5.8(a) and 5.8(b). It is obvious to see that the $\mathbf{C}^2$ configuration is more isotropic than the $\mathbf{C}^1$ configuration. However, for some specific points of attainable fore and torque spaces, the $\mathbf{C}^1$ configuration is larger than the $\mathbf{C}^2$ configuration.



(a) $\mathbf{C}^1$(red), $\mathbf{C}^2$(blue) (b) $\mathbf{C}^1$(red), $\mathbf{C}^2$(blue)

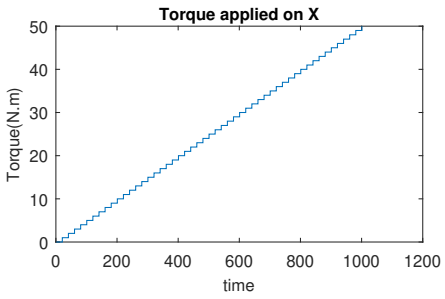
Figure 5.8 – Attainable spaces ((a)-Force space,(b)-Torque space) for two configurations ($\mathbf{C^1}$ (red) and $\mathbf{C^2}$(blue))

Thanks to the properties of matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ (Equation (5.1) and (5.2)) and the thruster characteristic (Figure 5.7), Table 5.6 shows the values of performance indices for two configurations. The performances of $\mathbf{C}^2$ configuration are better than ones of $\mathbf{C}^1$. Because of the calibration (the distance d_i is different between motors), the manipulability index (I_m) is larger than 1.

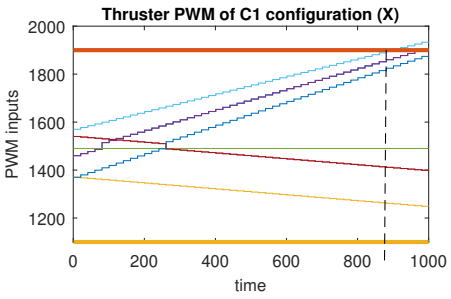
No.	Indices	$\mathbf{C}^1$	$\mathbf{C}^2$
1	I_m	7.12	2.559
2	I_e	3.32	2.09
3	I_w	6511536.45	10919428.13
4	I_{re}	4.05	1.56
5	I_{ro}	0	2

Table 5.6 – Comparison between two configurations (I_{ro} shows the maximum number of thrusters which can be failed to make sure that $rank(\mathbf{A}) = 6$)

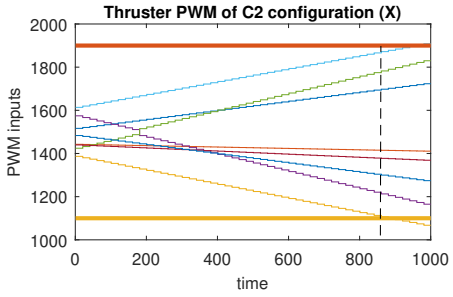
In order to verify the attainability of two configurations (workspace index), incremental torques are applied about X, Y, and Z axis respectively (Figures 5.9(a), 5.10(a), and 5.11(a)), the corresponding PWM (Pulse Width Modulation) inputs ($\mathbf{c}_m$) of 8 thrusters are computed. The results are shown in Figures 5.9(b), 5.9(c), 5.10(b), 5.10(c), 5.11(b), and 5.11(c) in which the two PWM's saturation values of thrusters (upper saturation value: 1900, lower saturation value: 1100) are plotted with two bold lines. We can see that the performances of the robot with two configurations are almost the same with the rotation about X and Y axis. However, the $\mathbf{C}^2$ configuration shows better performance with the rotation about Z-axis. In fact, the thrusters with $\mathbf{C}^1$ configuration reach saturations very earlier in comparison with the thrusters with $\mathbf{C^2}$ configuration (Figures 5.11(b) and 5.11(c)).



(a) Applied torque about X-axis (simulation)

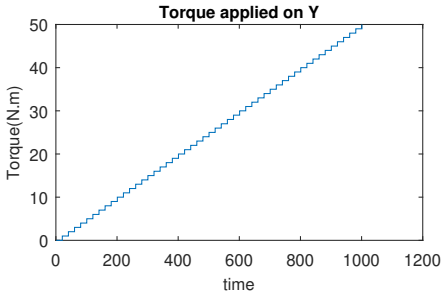


(b) PWM inputs of C1 (simulation)

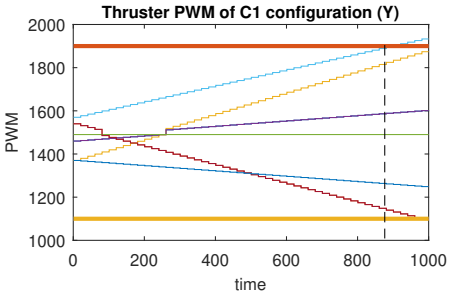


(c) PWM inputs of C2 (simulation)

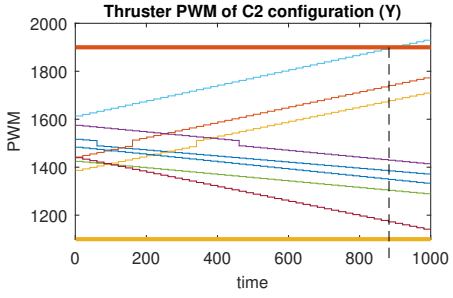
Figure 5.9 – The simulation of cube rotation about X-axis for $\mathbf{C}^1$ and $\mathbf{C}^2$



(a) Applied torque about Y-axis (simulation)

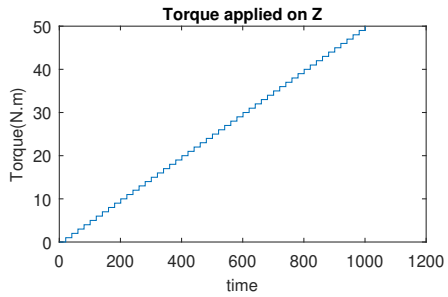


(b) PWM inputs of C1 (simulation)

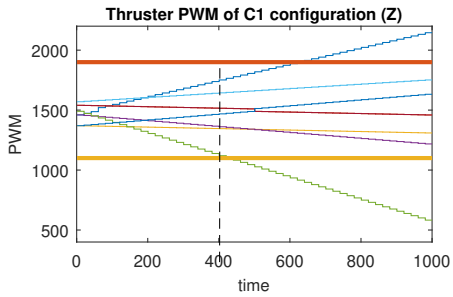


(c) PWM inputs of C2 (simulation)

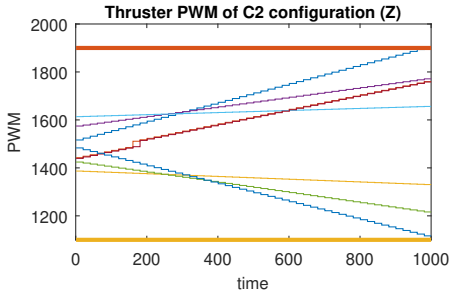
Figure 5.10 – The simulation of cube rotation about Y-axis for $\mathbf{C}^1$ and $\mathbf{C}^2$



(a) Applied torque about Z-axis (simulation)



(b) PWM inputs of C1 (simulation)



(c) PWM inputs of C2 (simulation)

Figure 5.11 – The simulation of cube rotation about Z-axis for $\mathbf{C}^1$ and $\mathbf{C}^2$

In order to validate the robustness of the optimal configuration ($\mathbf{C}^2$) in comparison with the normal configuration ($\mathbf{C}^1$), the rank of matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ is checked when arbitrary one or two columns have been nullified. When the resulting matrices are rank deficient, this means that the robustness is not guaranteed because one DoF is not actuated independently. Therefore, we can not control all 6 DOFs independently. The robustness index in Table 5.6 shows the checking results. In particular, when the 5^{th} thruster of $\mathbf{C}^1$ configuration fails, the robustness is not guaranteed.

The next sections present experimental results with these two configurations on real Cube robot.

5.2 Cube robot prototype

5.2.1 Descriptions of electronic and mechanic system

Cube robot is an over-actuated robot consisting of eight thrusters fixed at each corner of the cube and buoyancy pieces are fixed along its sides. Control and power electronic systems are set up in waterproof enclosures. It is controlled using quaternion formalism (see Chapter 2 and Appendix A.2). It can carry out many missions such as going straight, diving to predefined depth, and rotating around any axis. Cube robot is suitable to test our proposed approach. For the normal configuration, $\mathbf{C}^1$, thrusters are installed vertically or horizontally as in Figure 5.14(a). For

another configuration, $\mathbf{C}^2$, thruster'directions follow the optimal solution of our design problem as in Figure 5.14(b).

5.2.2 Cube's characteristics

This section summarizes the characteristics of Cube robot. Major components of the Cube are pressure sensor, electronic speed controller (ECS), thruster, and main card. Their technical details are shown in Table 5.7. The information diagram of Cube robot is displayed in Figure 5.12. DroPix card is the main processor in which IMU (gyroscope, accelerometer, and magnetometer) are integrated. It receives data from pressure sensor and controls thrusters through power switch card and electronic speed controller.

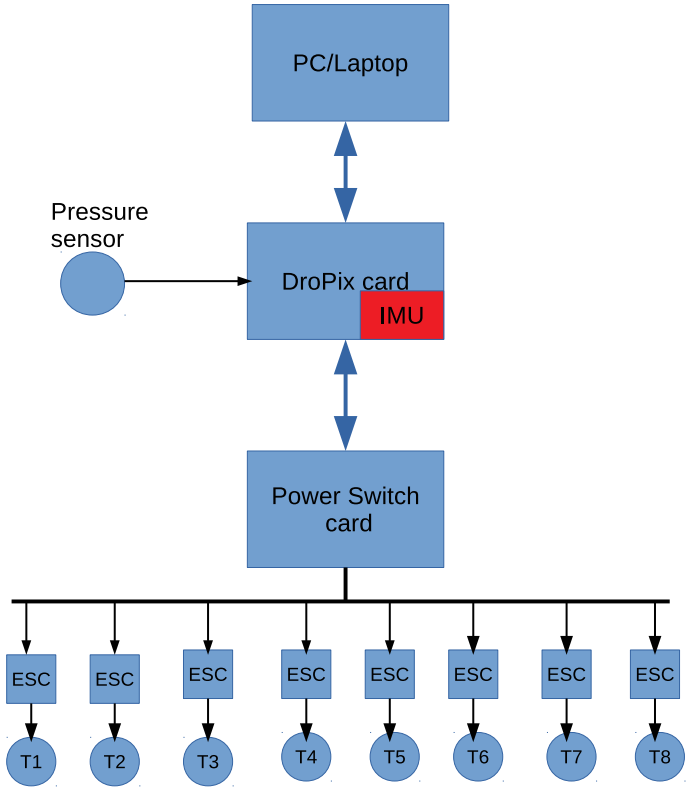


Figure 5.12 – Flow of information in Cube robot: T-Thruster, ESC-Electronic Speed Controller

5.3 Experimental results

Experiments are carried out on Cube robot to compare between two configurations, $\mathbf{C}^1$ (see Figure 5.14(a)), $\mathbf{C}^2$ (see Figure 5.14(b)), in the swimming pool at

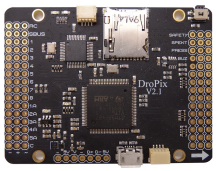
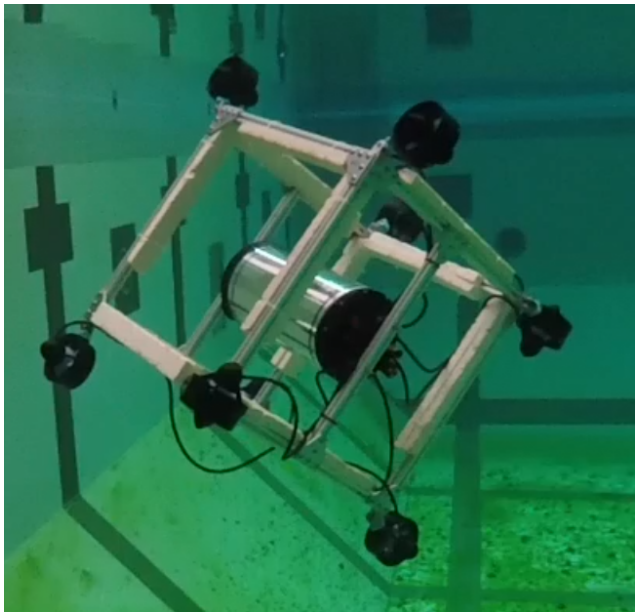
Devices	Images	Properties
Pressure sensor		— Depth resolution: 2mm — Communicate over I^2C — Measure up to 300m depth — Include a temperature sensor accurate to $\pm 1^0C$
ECS		— robust brushless electronic speed controller — Forward/reverse rotation direction
T200		— fully-flooded three-phase brushless motor — water-cooled motor — run at a range of voltages — use both clockwise and counter-clockwise propellers
DroPix V2		— Advanced 32 bit ARM Cortex — Communicate with additional peripherals (UART, I^2C, CAN) — Integrate gyroscope, accelerometer, and magnetometer

Table 5.7 – Technical details of main devices in Cube's robot

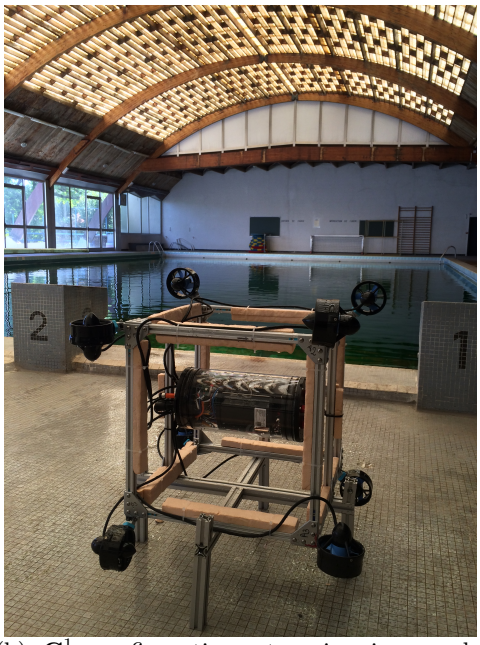
Montpellier University (see Figure 5.13(b)). The Cube in water and a video link for Cube's operations can be seen in Figure 5.13(a).

5.3.1 Attainability validation

Increasing torques about X-axis, Y-axis, and Z-axis are asked on Cube robot, angular velocities and PWM input values are stored for evaluating these two con-



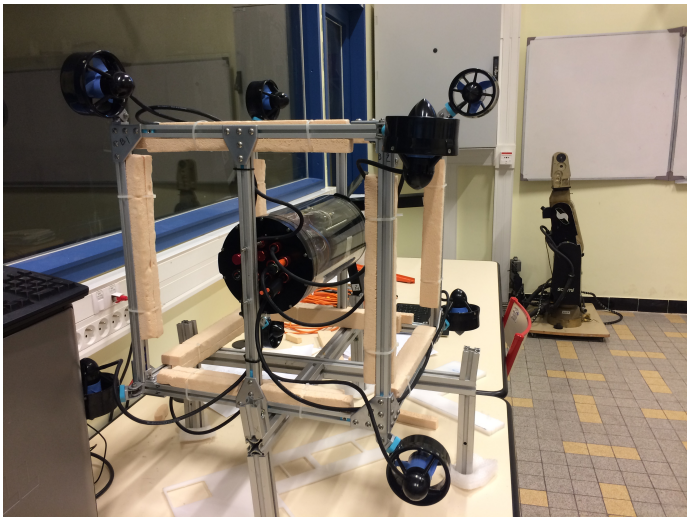
(a) Cubet robot in water https://www.youtube.com/watch?v=RKiWUOxDKdw



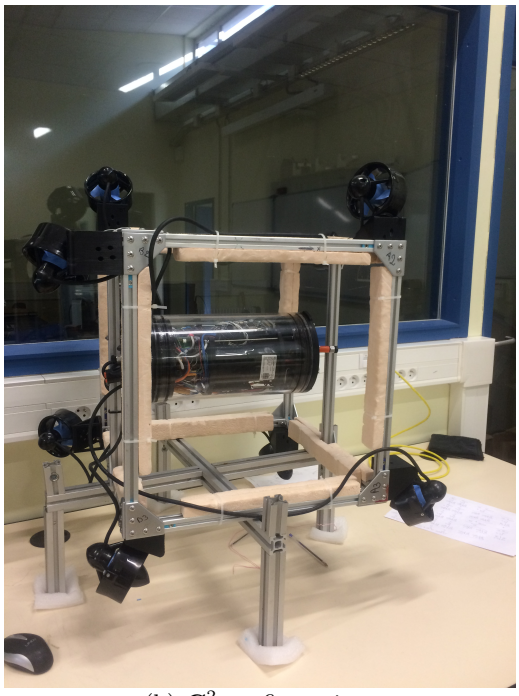
(b) $\mathbf{C}^1$ configuration at swimming pool, Montpellier University

Figure 5.13 – Experiments of Cube robot

figurations. For safety, the experiment will be stopped when one thruster reaches the saturation values. The experimental results are shown in Figures 5.15, 5.16 and 5.17. For rotating about X-axis, Figure 5.15, attainability of configurations $\mathbf{C}^1$ and $\mathbf{C}^2$ is almost the same, all thrusters operate in feasible region. Otherwise, for



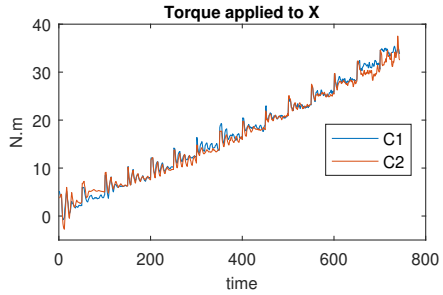
(a) $\mathbf{C}^1$ configuration



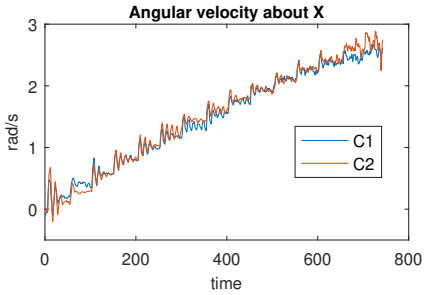
(b) $\mathbf{C}^2$ configuration

Figure 5.14 – $\mathbf{C}^2$ and $\mathbf{C}^1$ configurations

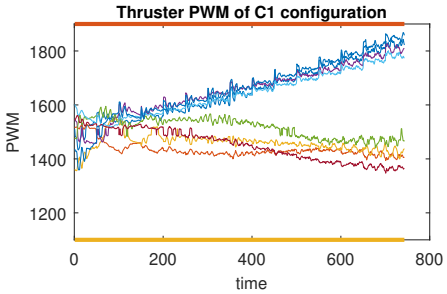
rotating about Y-axis and Z-axis, attainability of configuration $\mathbf{C}^2$ shows better $\mathbf{C}^1$ one. In particular, with Y-axis experiment (Figure 5.16), Cube robot with $\mathbf{C}^1$ stops the mission earlier than with $\mathbf{C}^2$ (at time step 771) because one thruster reach its saturation. The same behavior happens with Z-axis experiment (at time step 451) (see Figure 5.17).



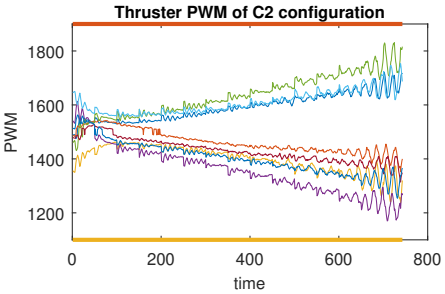
(a) Applied torque about X-axis



(b) Angular velocities

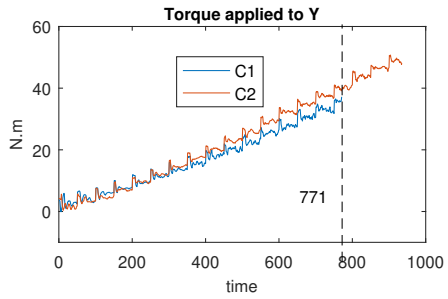


(c) PWM inputs of $\mathbf{C}^1$

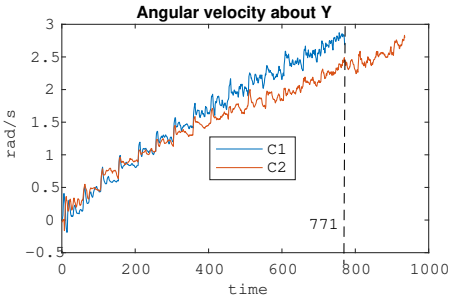


(d) PWM inputs of $\mathbf{C}^2$

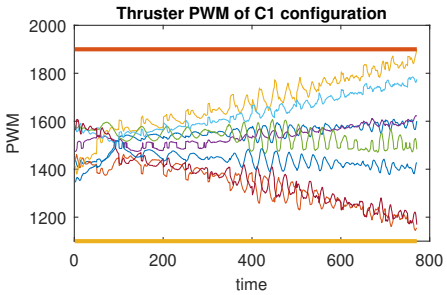
Figure 5.15 – The cube rotates about X-axis for $\mathbf{C}^1$ and $\mathbf{C}^2$



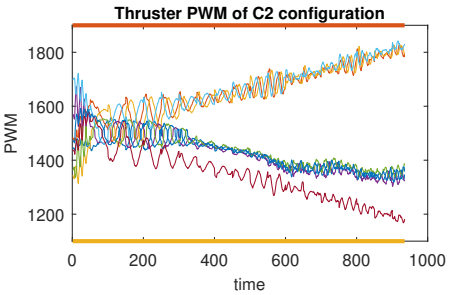
(a) Applied torque about Y-axis



(b) Angular velocities

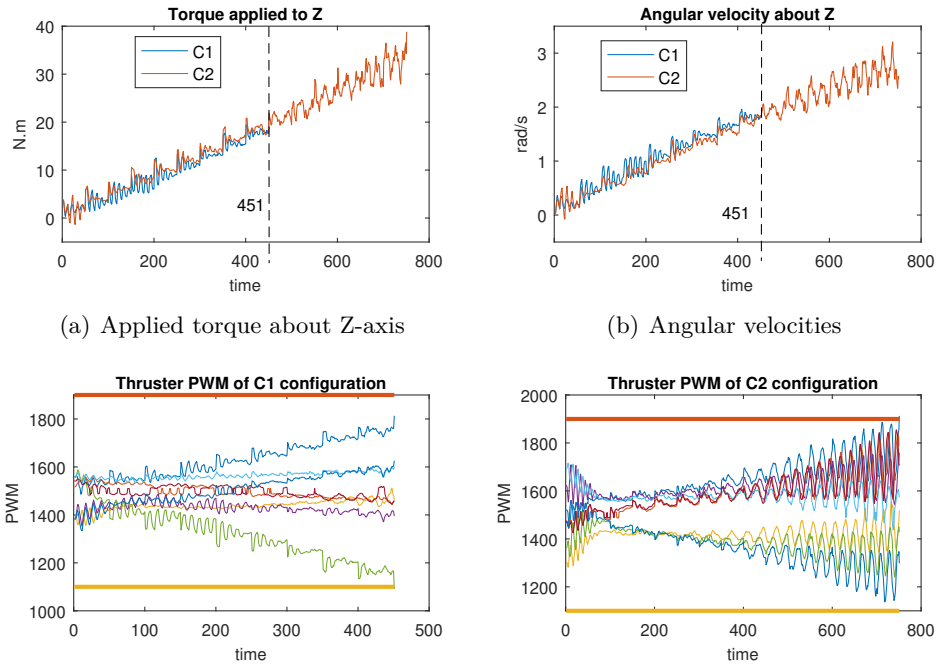


(c) PWM inputs of $\mathbf{C}^1$



(d) PWM inputs of $\mathbf{C}^2$

Figure 5.16 – The cube rotates about Y-axis for $\mathbf{C}^1$ and $\mathbf{C}^2$



(a) Applied torque about Z-axis

(b) Angular velocities

(c) PWM inputs of $\mathbf{C}^1$

(d) PWM inputs of $\mathbf{C}^2$

Figure 5.17 – The cube rotates about Z-axis for $\mathbf{C}^1$ and $\mathbf{C}^2$

5.3.2 Energetic validation

In this section, we verify the energy spending during these experiments for two configurations. An energy-like criterion is proposed:

$$\mathbf{E} = \sum_{i=1}^{m} \int_{t=0}^{T} |PWM^i(t) - 1500| dt \tag{5.3}$$

where m is the number of thrusters, T is the time of experiment, $PWM^i(t)$ is PWM inputs of i^{th} thruster.

Table 5.8 shows the energy-like consumption of robot during three rotations experiments. For X-axis rotation, the attainability of two configurations is the same but the the spent energy of $\mathbf{C}^2$ configuration is lower. For Y-axis and Z-axis rotation, the duration of experiments of $\mathbf{C}^2$ configuration is longer, the energy consumption, therefore, is higher.

No.	Rotation	$\mathbf{E}_{\mathbf{C}^1}$	$\mathbf{E}_{\mathbf{C}^2}$
1	X	7.2303e+04	6.9603e+04
2	Y	7.5480e+04	1.0590e+05
3	Z	3.1637e+04	7.4350e+04

Table 5.8 – Energy-like consumption of two configurations

Table 5.9 shows the comparison of energy-like consumption of two configurations with *the same time duration*. For Y-axis rotation, the energy value of $\mathbf{C}^2$ configuration is lower than one of $\mathbf{C}^1$ configuration. However, for Z-axis, the energy values of $\mathbf{C}^2$ configuration is higher. This happens because the robot dived deeper for $\mathbf{C}^2$ configuration experiments of Z-axis rotation, the robot had to deliver more power to keep at higher constant depth.

No.	Rotation	$\mathbf{E}_{\mathbf{C}^1}$	$\mathbf{E}_{\mathbf{C}^2}$
1	Y	7.5480e+04	7.2715e+04
2	Z	3.1637e+04	3.3312e+04

Table 5.9 – Energy consumption of two configurations with the same time duration

5.3.3 Robustness and Reactive validation

This section validates the robustness and reactive of the optimal configuration ($\mathbf{C}^2$) in comparison with the normal configuration ($\mathbf{C}^1$). For robustness, the robot performs a mission, and one or two thrusters is turned off. For the configuration $\mathbf{C}^1$, the mission will fail, and for the optimal configuration $\mathbf{C}^2$, the mission will continue. Specifically, for robustness index, we will carry out the following experiments:

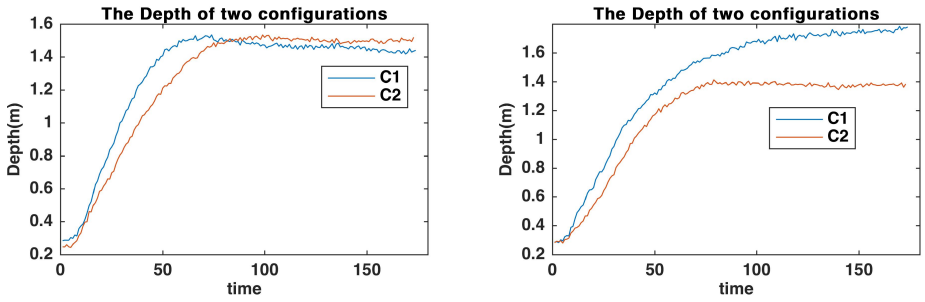
1. The Cube robot dives to predefined depth with all motors being in the normal operating conditions.
2. The Cube robot dives to the same predefined depth with one vertical motor being stopped.
3. The Cube robot dives to the same predefined depth with two vertical motors being stopped.
4. The Cube robot dives to the same predefined depth with three motors being stopped (two vertical motors and one arbitrary motor)
5. The Cube robot simultaneously dive to the same predefined depth and rotates about Z-axis with three motors being stopped (two vertical motors and one horizontal motor)

For reactive index, we will carry out the following experiments:

1. The Cube robot dives at the predefined depth and goes up to another predefined depth and dives again at the former predefined depth.
2. In the sequence, the Cube robot dives at the predefined depth, rotates about X-axis, after that, rotates about Y-axis. The rotation time of each axis should be 60 second or longer.
3. In the next, the Cube robot dives at the predefined depth, rotates about X-axis, after that, rotates about diagonal-axis (diagonal of the cube robot). The rotation time of each axis should be 60 second or longer.

The experimental results for robustness validation of $\mathbf{C}^1$ and $\mathbf{C}^2$ are shown in Figures 5.18, 5.19, and 5.20. In case of one or two motors stopped, the depth control

performance of $\mathbf{C}^1$ and $\mathbf{C}^2$ are almost the same (see Figure 5.18). The differences are clear in case of three thrusters stopped (Figure 5.20), the performance of $\mathbf{C}^1$ is not achieved (Figure 5.19) and violations of PWM values happen (see 5.20(a)).



(a) Depth control of two configurations with one motor stopped

(b) Depth control of two configurations with two motor stopped

Figure 5.18 – Depth control for $\mathbf{C}^1$ and $\mathbf{C}^2$ with one and two motors stopped

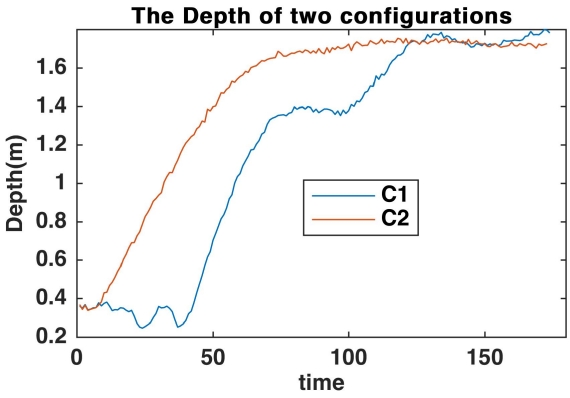
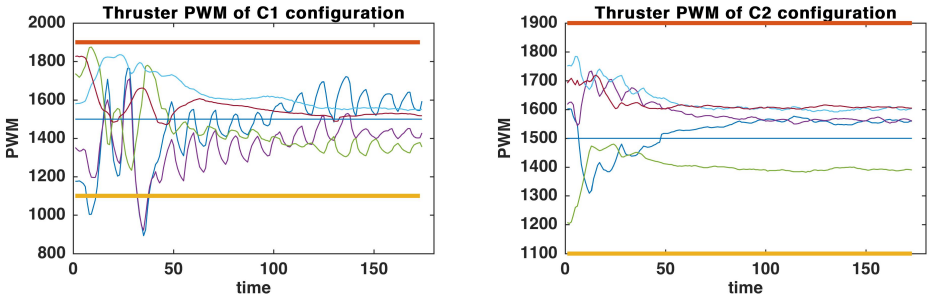


Figure 5.19 – Depth control for $\mathbf{C}^1$ and $\mathbf{C}^2$ with three motors stopped



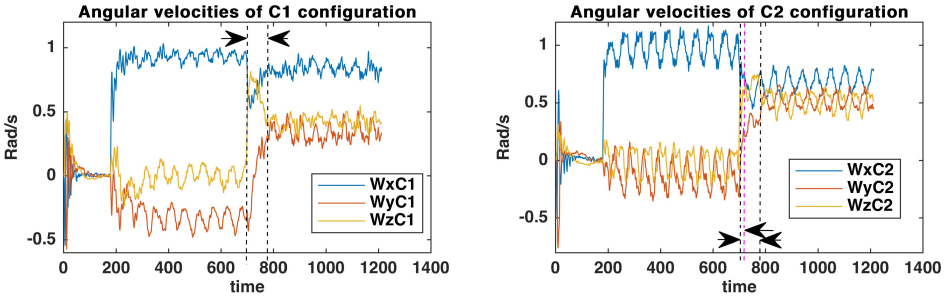
(a) PWM of $\mathbf{C}^1$ configuration

(b) PWM of $\mathbf{C}^2$ configuration

Figure 5.20 – PWM evaluation for $\mathbf{C}^1$ and $\mathbf{C}^2$ with 3 motors stopped

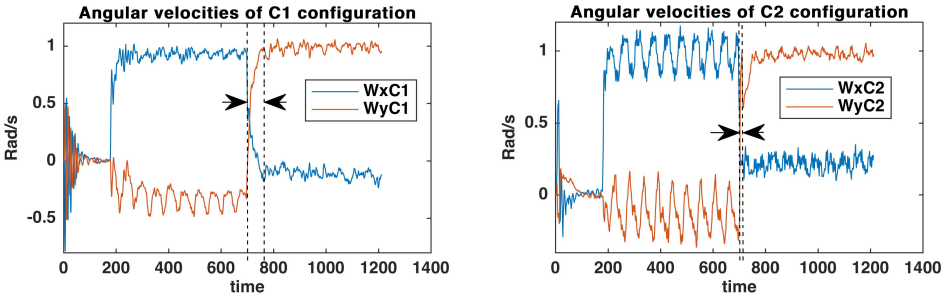
The results for reactive validation are shown in Figures 5.21, 5.22, and 5.23.

We measure the reactive time of angular velocities when changing the direction of Cube's actions. It is clear that reactive time of $\mathbf{C}^2$ configuration is faster than $\mathbf{C}^1$ configuration. Specifically, reactive time is the region formed by vertical lines in Figures 5.21, 5.22, and 5.23. It is obvious to see that reactive time of $\mathbf{C}^2$ configuration is smaller than one of $\mathbf{C}^1$ configuration (see Figures 5.22, and 5.23).



(a) Angular velocites of $\mathbf{C}^1$ configuration (b) Angular velocities of $\mathbf{C}^2$ configuration

Figure 5.21 – Angular velocity evaluation for $\mathbf{C}^1$ and $\mathbf{C}^2$: diving, rotating X-axis, and rotating diagonal-axis



(a) Angular velocites of $\mathbf{C}^1$ configuration (b) Angular velocities of $\mathbf{C}^2$ configuration

Figure 5.22 – Angular velocity evaluation for $\mathbf{C}^1$ and $\mathbf{C}^2$: diving, rotating X-axis, and rotating Y-axis

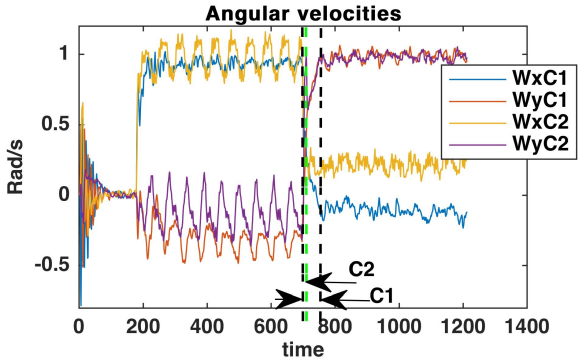


Figure 5.23 – Angular velocity evaluation for $\mathbf{C}^1$ and $\mathbf{C}^2$: diving, rotating X-axis, and rotating Y-axis

5.4 Conclusion

This chapter presented simulation and experiment results of static configuration problem in Chapter 4. The solving procedure was carried out to find a Pareto optimal configuration matrix in two case: general case (positions and directions of thrusters are unknown) and given-position case (positions of thrusters are known and given in corners of a cube). These solutions showed that performance indices were obtained except workspace index with attainment factor. A comparison between two configurations of given-position case, called $\mathbf{C^1}$ (normal configuration) and $\mathbf{C^2}$ (optimal configuration), was investigated in simulations and experiments. The first configuration is that thrusters are installed in vertical or horizontal directions, the second one is that thrusters are installed with optimal configuration from founded solution. Simulation results showed that performance indices of configuration $\mathbf{C^2}$ are better than ones of configuration $\mathbf{C^1}$.

A real robot (called Cube robot) was designed to validate the proposed method and quaternion formalism was used to design controller. Performances of optimal configuration ($\mathbf{C^2}$) showed better than ones of normal configuration ($\mathbf{C^1}$) in most of performance indices. For robustness index, the objective of mission with $\mathbf{C^1}$ configuration is also achieved, however, the behavior of the robot is worse than one of the robot with $\mathbf{C^2}$ configuration. Then robustness index was not satisfied in our definition. Note that the achievement of mission's objective not only depends on the configuration but also on control strategy and localization (path following or depth control with under-actuated system which can not be able to act along heave direction), and this is out of the thesis's scope.

Chapter 6

Reconfigurable Robot Design - Umbrella Robot

Contents

A prototype of an autonomous underwater robot with reconfigurable actuation system, called Umbrella robot, is built, and its properties are described. The reconfigurable capability is analyzed and acting abilities along/about 6 DoFs are proposed to evaluate a configuration. A configuration optimization problem with respect to geometric distances, in acting ability perspective, is suggested to enhance the robot's performances.

6.1 Introduction

In robotic fields, reconfigurable robots have been an attractive area because of their versatility. They can change their shape or configuration corresponding to specific mission requirements. Therefore, building cost can be reduced with one robot doing several works. Moreover, reconfigurable robots can be applied in complex tasks requiring adaptive configurations such as karst exploration or space applications. For instance, a clear operational reason of reconfigurable robots is to minimize power consumption. Robustness is also an advantage of reconfigurable robots in virtue of its flexibility. Readers can have the overview of questions and other issues of modular self-reconfigurable robot system in [Yim 2007] [Liu 2016].

In robot manipulators, the idea of reconfigurable robot appeared very early because of the development of manufacturing industry as shown in [Fukuda 1988a] [Fukuda 1988b] [Schmitz 1988] [Kereluk 2017]. This has been extended to other

fields in robotic domain such as land-based and underwater robotic areas. The prominent idea for reconfigurable robot is modular design concept in which the robot can connect or disconnect its corresponding modules [Murata 2007] [Stoy 2010]. For instance, a modular reconfigurable robot with perception-driven autonomy was proposed in [Daudelin 2018]. The robot is able to complete complex tasks by reactively reconfiguring to meet the perceived environmental information. Floor cleaning robot with reconfigurable mechanism was introduced in [Prabakaran 2018]. The robot reconfigures its morphology in response to its perceived environment to maximize coverage area. Reconfigurable snake robot was presented in [Thakker 2014]. The snake robot was also designed like modules. However, the robot can transform to various configurations without rearrangement of modules. Gait planner is used to switch between gaits: snake gait, transforming gait, and walking gait.

In the underwater field, a guidance and control method for a reconfigurable unmanned underwater vehicle was introduced in [Caccia 2000]. However, the reconfigurability of the robot is not clear and all thrusters were installed hardly. In [Low 2007], a reconfigurable robotic fish with undulating fins was developed. But, it is not a reconfigurable capability during operation, just reconfiguring design parameters to achieve an another version of the robot. Another reconfigurable robotic fish was introduced in [Hu 2007]. Tcomputinghe robot was designed in the way of modules combination. This can help to build robotic fish with different morphologies. So, this is just a static reconfiguration. Reconfigurable magnetic-coupling thrusters for AUVs was introduced in [Chocron 2008][Chocron 2013][Vega 2016] and [Chocron 2018]. The main idea here is based on the coupling of two magnetic parts. One servo motor is used to change the relative position of one magnet with respect to another one. Therefore, the more thrusters are in the robot, the more servo motor to be actuated. This yields the cost of robot being increased. Moreover, the magnetic filed between coupling magnets is easily disturbed by metal parts of the robot structure and the force from motor rotor to propeller robot is lost. The idea using magnetic coupling to build versatile thruster configuration is also used in [Pugi 2018]. Reconfigurable AUV for Intervention (RAUVI project) was presented in [De Novi 2009] [Sanz 2010] [Prats 2012]. This is an autonomous underwater robot equipped with one manipulator which allows to perform manipulation tasks. The robot, inherited from Girona 500 AUV [Ribas 2011], is statically reconfigured with respect to different tasks. A prototype of a reconfigurable underwater robot with bioinspired electric sense was introduced in [Mintchev 2012]. The robot was designed as modules which can be detached or attached in order to adapt its configuration. In [Meister 2013], dynamics and a control approach for modular and self-reconfigurable robotic systems were presented. Several benchmark examples are used to evaluate different configurations. In robotic systems, the reconfigurability can be found in any stages of the robotic architecture from software to hardware, concepts of reconfigurable autonomy can be found in [Dennis 2014]. A static reconfigurable underwater robot, namely *SeaDrone*, was introduced [Moreno 2014]. Four configurations of the robot corresponding with four underwater tasks were shown. However, this is done statically before starting missions. A structure of a

reconfigurable AUV/ROV for man-robot underwater cooperation was depicted in [Odetti 2017]. This is actually a ROV and is possible to change it into an AUV. Moreover, it can be mechanically changed to six possible layouts.

6.2 Principles

6.2.1 General view

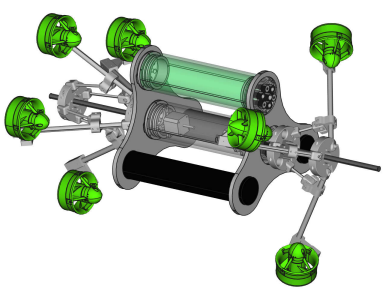
The first idea of Umbrella Robot stems from [Gourmelen 2018]. The general view of the Umbrella Robot (UR) is shown in Fig 6.1 in which robot is shown to be able to modify its configuration, i.e., forward thruster in *"open"* state in Figure 6.1(a), *"close"* state in Figure 6.1(b), and *"open-open"* state (forward and backward sides) in Figure 6.1(c). The robot carries seven thrusters, four thrusters in **backward side** and three thrusters in **forward side**. The body of UR is built by two tubes in which one is used for central processor, the other holds two DC motors connected to two threaded rods (forward and backward sides) are used to change configurations. Two battery packages (black tubes in Figure 6.1) are used for the robot. Waterproof cables are used for communicating between parts of the robot.

6.2.2 Hardware Architecture

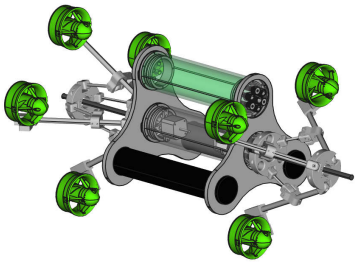
We portrait the hardware architecture of the UR in this section. The processor of UR is Raspberry Pi 2. Raspberry Pi has many merits in computational capability and extensibility. The robot is equipped with a pressure sensor and an IMU sensor. One camera and GPS module will be installed in the future. Thrusters of the robot are controlled by a PWM module which communicates with Raspberry Pi by *I2C* protocol. Two DC motors with encoders are used to change the orientations and positions of thrusters. Communications between the Umbrella Robot and a PC/lapop is wireless protocol. The principle architecture of UR is illustrated in Figure 6.2. Electronic Speed Controller (ESC) is provided for each thruster. Two DC motors are controlled by DC motor drivers. Battery packages with power converter card can supply electric power with several voltage levels for the whole robot.

6.2.3 Software architecture

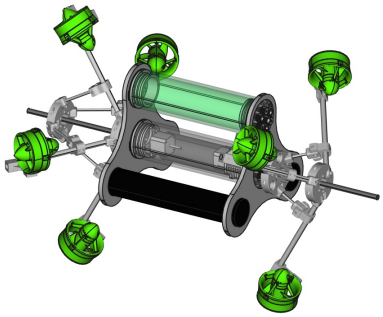
This section presents the software architecture of the robot from use case diagram to dynamic state machine modeling. Because of the limitation of pages, some diagrams of software design process are not illustrated, for instance, dynamic interaction modeling, integrated communication diagrams.



(a) Umbrella robot in open-forward



(b) Umbrella robot in close



(c) Umbrella robot in open-forward and open-backward

Figure 6.1 – The 3D model of Umbrella Robot

6.2.3.1 Use case diagram

Use case diagram depicts all operation cases of the robot in practical uses. The use case diagram is illustrated in Figure 6.3. For starting missions, an operator *turns on power* of the robot, automatic or manual operation, is chosen based on parameters setting up in the UR'software. The *turn off* use case is carried out when finishing missions or stopping the robot for reasonable issues. *Upload data* use case is activated when an operator wants to store all data or states of the current missions.

6.2.3.2 Static and object/class modeling

Object structuring diagram is shown in Figure 6.4. The software system includes four input classes, two output classes, a proxy class, two entity classes, and one state dependent class. The input classes, IMU, pressure sensor, and two encoders, receive sensory data from sensors. The output classes including DC motors and thrusters interface communicate with DC motors and thrusters. Two entity classes, UMSetZero and UmRData, are used to set up zero point (for changing

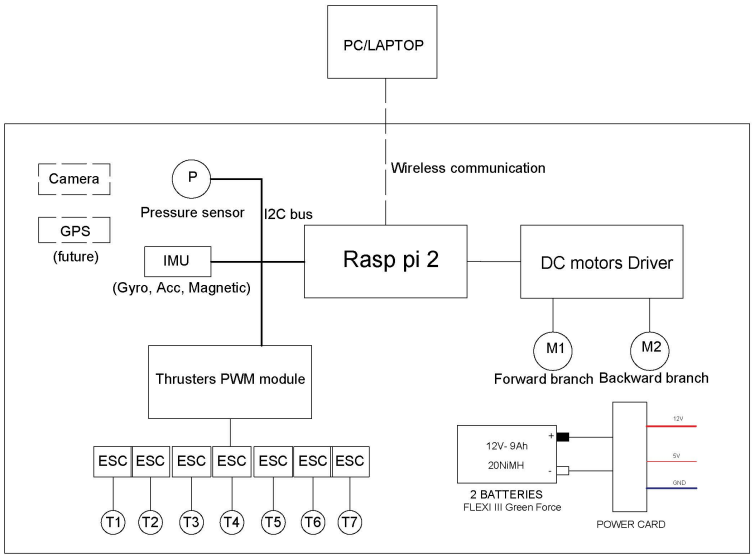


Figure 6.2 – The principle diagram of UR

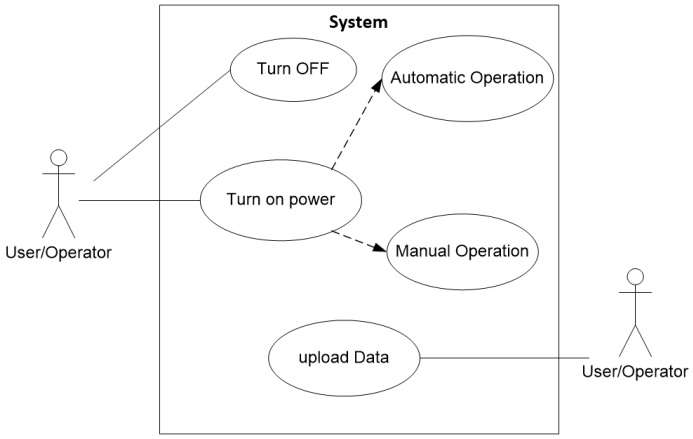


Figure 6.3 – The use case diagram of UR

UR's configurations) and to store operation data respectively. One proxy class transfers data to external devices such as personal computer or laptop.

6.2.3.3 Dynamic state machine modeling

The dynamic state machine diagram of state dependent class describes states of the robot during operations. The state machine follows the states of the robot as its transitions from idle state to other states. The states are determined by following the use cases (Turn on power/automatic, turn on power/manual, turn off, upload data). In the paper, only dynamic state machine of *Turn on power/automatic* use case is shown as follows:

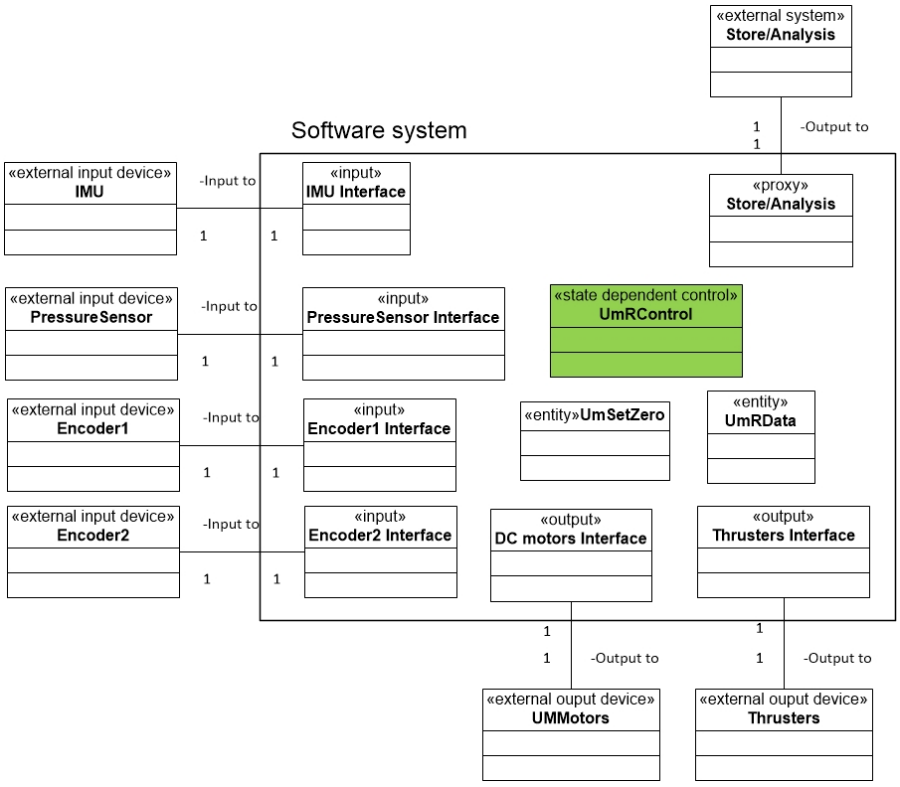


Figure 6.4 – The object structuring diagram of UR

1. **Idle**: This is the initial state, in which the robot is idle, waiting for a specific time before starting missions (this time is saved for putting the robot into the water). In this state, the robot sets umbrella into zero point and checks all initial conditions.
2. **Starting**: This state is entered when the waiting time of *Idle* state is elapsed and start command is sent to thrusters.
3. **Moving**: The robot is moving or rotating or keeping position. This depends the control strategy.
4. **ChangeConfig**: This state is active when the robot changes its configurations. In this state, the robot just varies the positions and orientations of thrusters, not doing anything else.
5. **Moving/ChangeConfig**: The robot enters this state when a command of changing configuration is received. The robot is still doing current missions and changing its configurations.
6. **Stopping**: The robot enters this state when finishing missions.

Figure 6.5 portraits the transitions between states. The notation *condition/action* is used to describe the transition arrows.

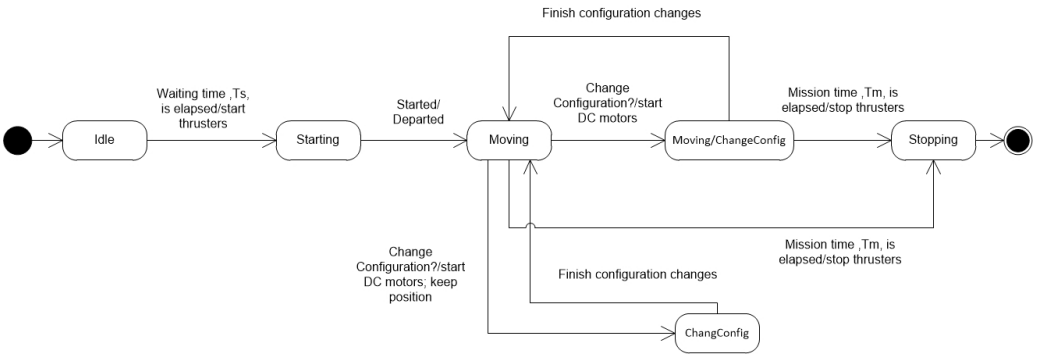


Figure 6.5 – The Dynamic state machine modeling of *Turn on power/automatic* use case

6.3 Reconfigurability

The reconfigurability of the robot is expressed by varying two angles α_F and α_B (see Figure 6.6). Therefore, the configuration matrix, the $\mathbf{A}$ matrix will change accordingly. The relation between thrusters forces and resulting forces (forces/torques w.r.t body frame) is usually as follows:

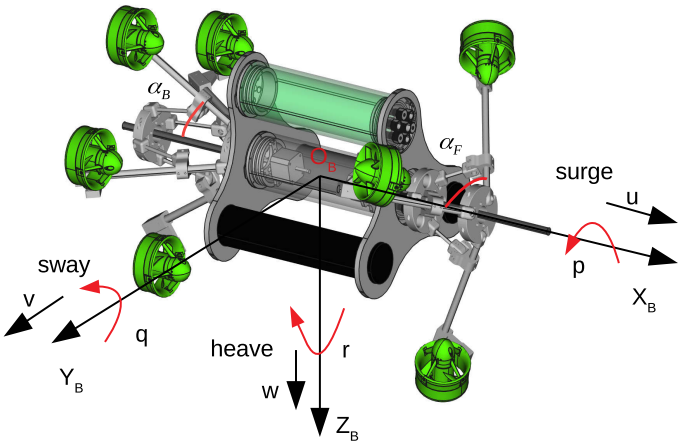


Figure 6.6 – Definitions of two angles α_F and α_B

$$\mathbf{F}_B = \mathbf{A}(\alpha_F, \alpha_B)\mathbf{F}_m = \begin{pmatrix} \mathbf{F} \\ \mathbf{\Gamma} \end{pmatrix} \tag{6.1}$$

where $\mathbf{F}_B = [F_u \quad F_v \quad F_w \quad \Gamma_p \quad \Gamma_q \quad \Gamma_r]^T \in \mathbb{R}^6$ is vector of resulting force in which $\mathbf{F} = [F_u \quad F_v \quad F_w]^T$ and $\mathbf{\Gamma} = [\Gamma_p \quad \Gamma_q \quad \Gamma_r]^T$, $\mathbf{A}(\alpha_F, \alpha_B) \in \mathbb{R}^{6\times m}$, and thruster

force vector $\mathbf{F}_m = [F_{m,1} \quad F_{m,2} \quad ... \quad F_{m,m}]^T \in \mathbb{R}^m$ is vector of thrusters forces, and m is the number of thrusters, $m = 7 > 6$. From the scheme of Umbrella Robot, the configuration matrix, $\mathbf{A}$, has a form as:

$$\begin{aligned}\mathbf{A} &= \begin{pmatrix} \mathbf{u}_1^B & \mathbf{u}_2^B & \cdots & \mathbf{u}_m^B \\ \mathbf{r}_1^B \otimes \mathbf{u}_1^B & \mathbf{r}_2^B \otimes \mathbf{u}_2^B & \cdots & \mathbf{r}_m^B \otimes \mathbf{u}_m^B \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{u}_1^B & \mathbf{u}_2^B & \cdots & \mathbf{u}_m^B \\ \boldsymbol{\tau}_1^B & \boldsymbol{\tau}_2^B & \cdots & \boldsymbol{\tau}_m^B \end{pmatrix} = \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{pmatrix}\end{aligned} \tag{6.2}$$

where $m = 7$; $\mathbf{u}_i^B$ and $\mathbf{r}_i^B$ $(i \in 1, ..., 7)$ are directions and positions of thrusters w.r.t body frame. The operator $\otimes$ is cross product, and $\boldsymbol{\tau}_i^B = \mathbf{r}_i^B \otimes \mathbf{u}_i^B$

The $\mathbf{u}_1^B, \ldots, \mathbf{u}_7^B$ and $\mathbf{r}_1^B, \ldots, \mathbf{r}_7^B$ are identified as in Appendix A.3. The geometries for this computation are shown in Figure A.1 and Figure A.2.

From the equations $\mathbf{u}_i^b$ and $\mathbf{r}_i^b$ in Appendix, it is obvious to see that the configuration matrix, $\mathbf{A}$, depends on forward and backward angles, α_F and α_B, respectively (we can also see these angles geometrically in Figure A.1 and Figure A.2). By varying these two angles, the configuration matrix will be changed. Table 6.1 shows the configuration matrix $\mathbf{A}$ corresponding with three cases. If $\alpha_F = \alpha_B = 45°$, the robot is like a Torpedo robot, and it can act along heave, pitch, and yaw directions. This configuration corresponds to an under-actuated situation, and the system can be controlled as a torpedo-like system. Otherwise, if $\alpha_F = \alpha_B = 90°$, the robot can act along 6 DoFs, it is a fully actuated system (note that in roll direction, the acting ability is quite small). Therefore, the acting capability of robot is extended. In arbitrary case, for instance $\alpha_F = 50°, \alpha_B = 60°$, the configuration matrix also shows that the robot can operate along 6 DoFs, however, the priority is also along u axis, and then the robot can be considered as a fully actuated system, but with different capability along specific DoF. It is clear to see this in the simulation results. If two angles α_F and α_B vary at each time step, the online adaption of configuration matrix can be achieved. This will be shown and discussed more in Chapter 7 and Chapter 8.

6.4 Prototype

The current version of Umbrella Robot at LIRMM, Montpellier University, is shown in Figure 6.7. For varying configuration, readers can follow a video link in Figure 6.7

6.5 Configuration evaluation - Acting ability

This section proposes a criterion to evaluate the acting ability along each DoFs of robot based on configuration matrix. Configuration matrix, $\mathbf{A}$ has a general form

Angles	**A** matrix
$\alpha_F = 45°$ $\alpha_B = 45°$	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0435 & 0.1909 & -0.0435 & -0.0600 & 0.1909 & -0.0600 & -0.1909 \\ 0.1953 & -0.0600 & -0.1953 & 0.1909 & -0.0600 & -0.1909 & 0.0600 \end{pmatrix}$
$\alpha_F = 50°$ $\alpha_B = 60°$	$\begin{pmatrix} 0.9962 & 0.9962 & 0.9962 & 0.9659 & 0.9659 & 0.9659 & 0.9659 \\ 0.0755 & 0 & -0.0755 & 0.2588 & 0 & -0.2588 & 0 \\ 0.0436 & -0.0872 & 0.0436 & 0 & -0.2588 & 0 & 0.2588 \\ -0.0052 & -0.0052 & 0.0052 & 0.0155 & -0.0155 & -0.0155 & -0.0155 \\ -0.0630 & 0.2296 & -0.0630 & -0.0580 & 0.1417 & -0.0580 & -0.1417 \\ 0.2287 & -0.0598 & -0.2287 & 0.1417 & -0.0580 & -0.1417 & 0.0580 \end{pmatrix}$
$\alpha_F = 90°$ $\alpha_B = 90°$	$\begin{pmatrix} 0.7071 & 0.7071 & 0.7071 & 0.7071 & 0.7071 & 0.7071 & 0.7071 \\ 0.6124 & 0 & -0.6124 & 0.7071 & 0 & -0.7071 & 0 \\ 0.3536 & -0.7071 & 0.3536 & 0 & -0.7071 & 0 & 0.7071 \\ -0.0424 & -0.0424 & 0.0424 & 0.0424 & -0.0424 & -0.0424 & -0.0424 \\ -0.1575 & 0.3885 & -0.1575 & -0.0424 & 0.0566 & -0.0424 & -0.0566 \\ 0.3577 & -0.0424 & -0.3577 & 0.0566 & -0.0424 & -0.0566 & 0.0424 \end{pmatrix}$

Table 6.1 – configuration matrix with some cases of two angles α_F and α_B

Figure 6.7 – A prototype of Umbrella Robot https://www.youtube.com/watch?v=yBBCu1z3q-0&feature=youtu.be

as:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix} \tag{6.3}$$

The acting abilities along each DoFs belongs to elements of **A** matrix and are

defined as:

$$a_u = \sum_{k=1}^{m} a_{1k}^2 \tag{6.4}$$

$$a_v = \sum_{k=1}^{m} a_{2k}^2 \tag{6.5}$$

$$a_w = \sum_{k=1}^{m} a_{3k}^2 \tag{6.6}$$

$$a_p = \sum_{k=1}^{m} a_{4k}^2 \tag{6.7}$$

$$a_q = \sum_{k=1}^{m} a_{5k}^2 \tag{6.8}$$

$$a_r = \sum_{k=1}^{m} a_{6k}^2 \tag{6.9}$$

where $a_u, a_v, a_w, a_p, a_q, a_r$ are acting abilities along u, v, w and about p, q, r in body-frame respectively, under the condition that $rank(\mathbf{A}) = 6$.

For our Umbrella Robot, two angles α_F, α_B vary from 45^0 to 90^0. We can illustrate variations of these capabilities as in Figure 6.8. We can obviously choose the maximum value of acting ability of each DoF. However, there exists a large deviation of acting abilities between DoFs. In particular, acting abilities of 6 DoFs with $\alpha_F = \alpha_B = 90^0$ are shown in Figure 6.9. It is obvious to see that, about p-axis (roll direction), the acting ability of robot is too small. This is a disadvantage of the robot's current version

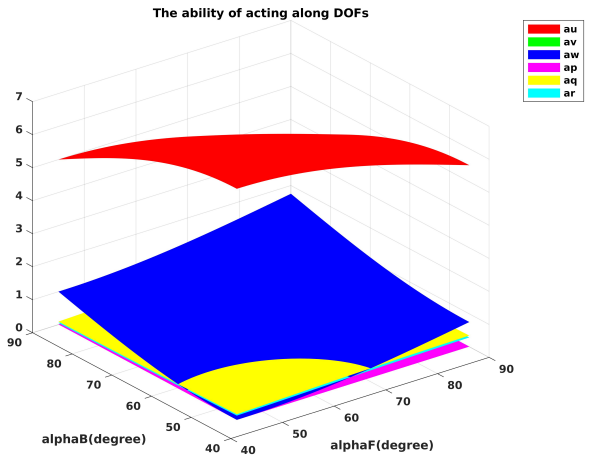


Figure 6.8 – Acting abilities along/about each DOFs of Umbrella robot with varying α_F and α_B

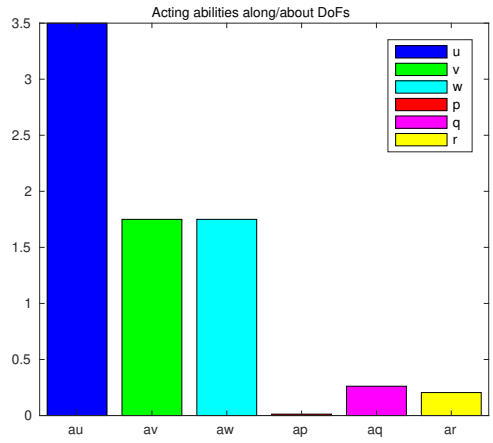


Figure 6.9 – Acting abilities along/about each DOFs of Umbrella robot with $\alpha_F = \alpha_B = 90^0$

We propose a configuration optimization problem, in which the deviations between u, v, w and p, q, r are minimized in the next section.

6.6 Configuration optimization - Acting ability

This section presents configuration optimization problem with respect to geometric distances and initial angles of thrusters. The objective function is:

$$J = \sum w_i \delta_i^2 \tag{6.10}$$

where $i = \{u, v, w, p, q, r\}$, $\delta_i = a_i - m, i = \{u, v, w, p, q, r\}$ is the deviation of acting ability of i^{th} DOF (a_i) and acting average value $(m = \sum a_i/6)$, and w_i is corresponding weight. See Figure 6.10 for more details.

The problem is formulated as:

$$\min_{\mathbf{x}} J = \sum w_i \delta_i^2, i = \{u, v, w, p, q, r\} \tag{6.11}$$

$$s.t \tag{6.12}$$

$$\underline{\alpha_F} \leq \alpha_F \leq \overline{\alpha_F} \tag{6.13}$$

$$\underline{\alpha_B} \leq \alpha_B \leq \overline{\alpha_B} \tag{6.14}$$

$$0 < d \leq \overline{d} \tag{6.15}$$

$$0 < d_e \leq \overline{d_e} \tag{6.16}$$

$$0 < d_t \leq \overline{d_t} \tag{6.17}$$

$$0 < d_c \leq \overline{d_c} \tag{6.18}$$

$$0 < d_b \leq \overline{d_b} \tag{6.19}$$

$$\underline{az_i} \leq az_i \leq \overline{az_i}, i \in \{1, ..., 7\} \tag{6.20}$$

$$\underline{el_i} \leq el_i \leq \overline{el_i}, i \in \{1, ..., 7\} \tag{6.21}$$

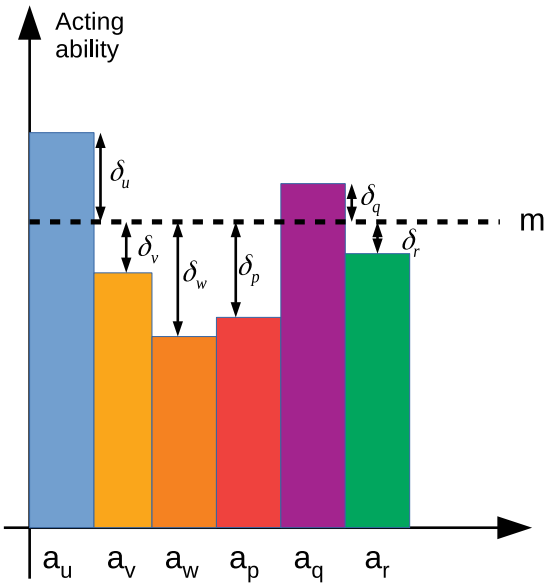


Figure 6.10 – Acting ability along each DOFs and deviations

where $\mathbf{x} = [\alpha_F \quad \alpha_B \quad d \quad de \quad d_t \quad d_c \quad d_b \quad az_i \quad el_i]^T$, and $\underline{()}$ and $\overline{()}$ are minimum and maximum value of each variable respectively. See Figure 6.11 for more details.

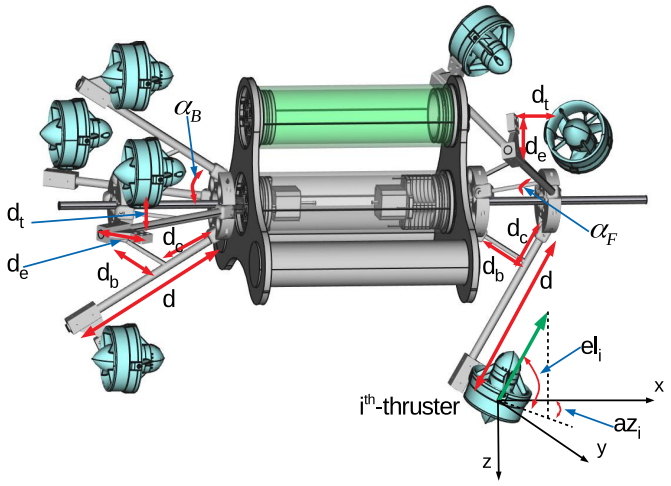


Figure 6.11 – variables in configuration optimization problem

Global search method is used to find a solution. The limitations of variables are presented in Table 6.2. The optimal results are shown in Table 6.3.

Optimal configuration matrix **A** is shown in Equation 6.22 and acting abilities of this optimal configuration are shown in Figure 6.12(a). A comparison of acting abilities between current configuration and optimal one (w.r.t geometric distances)

No.	Variable	Lower limit	Upper limit
1	α_F	$45°$	$90°$
2	α_B	$45°$	$90°$
3	d	$0(m)$	$0.5(m)$
4	d_e	$0(m)$	$0.2(m)$
5	d_t	$0(m)$	$0.2(m)$
6	d_c	$0(m)$	$0.15(m)$
7	d_b	$0(m)$	$0.15(m)$
8	az_i	$-\pi$	π
9	el_i	$-\pi$	π

Table 6.2 – Limitations of variables

No.	Variable	Optimal value
1	α_F	$90°$
2	α_B	$45°$
3	d	$0.5(m)$
4	d_e	$0.2(m)$
5	d_t	$0.2(m)$
6	d_c	$0.0914(m)$
7	d_b	$0.15(m)$
8	az_i, el_i(rad)	$\begin{pmatrix} az_1 = 0.6106 & el_1 = 2.2317 \\ az_2 = 1.7060 & el_2 = 0.4642 \\ az_3 = -0.6106 & el_3 = -0.9099 \\ az_4 = -0.3754 & el_4 = 2.6030 \\ az_5 = 0.1261 & el_5 = -2.5692 \\ az_6 = 2.9525 & el_6 = -1.9817 \\ az_7 = 3.0155 & el_7 = 0.5724 \end{pmatrix}$

Table 6.3 – Optimal values

is given in Figure 6.12(b). It is clear to see that the deviation of acting abilities between 6 DoFs of optimal configuration is smaller than ones of current configuration.

$$\mathbf{A} = \begin{pmatrix} -0.2081 & 0.1279 & -0.2081 & -0.7987 & 0.8339 & -0.3923 & 0.8339 \\ -0.8275 & -0.9402 & 0.8275 & 0.3149 & 0.1057 & 0.0751 & -0.1056 \\ 0.5215 & 0.3157 & 0.5215 & 0.5128 & 0.5417 & 0.9168 & -0.5417 \\ -0.5239 & 0.5332 & 0.5239 & -0.1908 & 0.0560 & 0.4688 & 0.0561 \\ -0.1929 & -0.1062 & -0.1930 & 0.3813 & 0.6468 & 0.4746 & -0.6469 \\ -0.5153 & -0.5323 & 0.5152 & -0.5314 & -0.2125 & 0.1617 & 0.2124 \end{pmatrix} \tag{6.22}$$

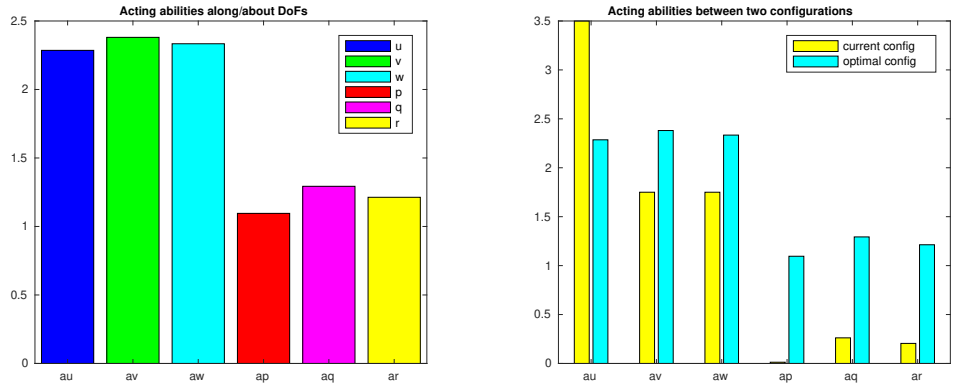
A summary of performance index values and acting abilities for this new configuration is shown in Table 6.4

No.	Performance Indices and Acting abilities	Value
1	I_m	4.56
2	I_e	2.48
3	I_w	$7.16.10^6$
4	I_{re}	2.29
5	I_{ro}	1
6	a_u	2.28
7	a_v	2.38
8	a_w	2.34
9	a_p	1.10
10	a_q	1.29
11	a_r	1.21

Table 6.4 – Performance indices and acting abilities of optimal configuration

6.7 Conclusion

Reconfigurable robots have been attracted many researchers in recent years because of amazing mechanic system and flexibility. This chapter presented the procedure to build a reconfigurable configuration robot, also called Umbrella Robot in which robot's configuration depends on two angles (α_F - angle of front branch of thrusters, and α_B - angle of rear branch of thrusters). The robot consists of 7 thrusters and their positions and directions can be varied. The main processor is a small computer, *Pi-2*, which can easily extend to complex task such as image processing, SLAM processing. When $\alpha_F = \alpha_B = 45^0$, the robot is like a torpedo-shape robot which has maximum ability in surge direction, otherwise, if $\alpha_F = \alpha_B = 90^0$, robot can act along 6 DoFs. Moreover, in this chapter, acting abilities of a configuration matrix was proposed. The fact that current Umbrella Robot also



(a) Optimal acting abilities along/about each DoFs of Umbrella robot

(b) A comparison with current configuration

Figure 6.12 – Optimal acting abilities along/about each DoFs of Umbrella robot and the comparison with current configuration

shows some limitations, for instance, the acting ability about roll is quite small, the deviations between DoFs remain large. In order to optimize configuration design, an optimization problem with respect to geometric distances and initial angles of thrusters was suggested. A founded solution showed an advanced configuration in which the acting abilities along $u/v/w$ and about $p/q/r$ are almost the same.

Up to this chapter, we proposed performance indices and acting abilities of an underwater robot (through configuration matrix) and we designed two real robots (Cube robot and Umbrella robot) with different configurations. An interesting summary of these issues are presented in Table 6.5. This is realized by a Matlab-based Toolbox, which can be referred to Appendix D for more details. Following Table 6.5, we can see that Cube robot with configuration $\mathbf{C^2}$ shows better performances than its configuration $\mathbf{C^1}$. A glance of comparison between Ball robot and *SamoS* can be considered. It is obvious to see that Ball robot, carrying 8 thrusters, possesses better acting abilities and several performance indices than *SamoS*. When Ball robot carries 6 thrusters, its performances are the same as *SamoS*.

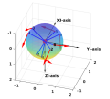
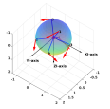
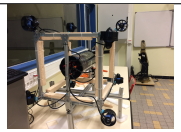
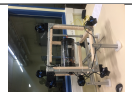
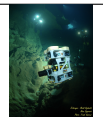
No.	Photo	Name	I_m	I_e	I_w	I_{re}	I_{ro}	a_u	a_v	a_w	a_p	a_q	a_r
1		Ball robot (8 thrusters)	1.00	1.22	20307161.00	0.61	2	2.66	2.66	2.66	2.66	2.66	2.66
2		Ball robot (6 thrusters)	1.00	1.41	8268848.73	0.71	0	2.00	2.00	2.00	2.00	2.00	2.00
2		SamoS [Pierrot 1998]	1.00	1.41	8445163.62	0.70	0	2.00	2.00	2.00	2.00	2.00	2.00
4		Cube ($\mathbf{C^1}$)	7.12	3.32	6511536.45	4.05	0	2.00	3.00	3.00	0.43	0.36	0.36
5		Cube ($\mathbf{C^2}$)	2.55	2.09	10919428.13	1.56	2	2.66	2.66	2.66	0.42	0.54	0.45
6		UmRobot	$\begin{bmatrix}42.35\\ \infty\end{bmatrix}$	$\begin{bmatrix}2.32\\ 4255.2\end{bmatrix}$	$\begin{bmatrix}5215.4\\ 5.6287.10^6\end{bmatrix}$	$\begin{bmatrix}3.09\\ 7.14.10^3\end{bmatrix}$	$\begin{bmatrix}0\\ 1\end{bmatrix}$	$\begin{bmatrix}3.5\\ 7.0\end{bmatrix}$	$\begin{bmatrix}0\\ 1.75\end{bmatrix}$	$\begin{bmatrix}0\\ 1.75\end{bmatrix}$	$\begin{bmatrix}0\\ 0.012\end{bmatrix}$	$\begin{bmatrix}0.09\\ 0.34\end{bmatrix}$	$\begin{bmatrix}0.05\\ 0.28\end{bmatrix}$
7		Ulysse	6.15	2.60	19423247.33	2.51	0	6	2	4	0.16	0.24	0.41

Table 6.5 – Performance indices and acting abilities of different robots, it shows [min max] in case of UmRobot, I_{ro} shows the maximum number of thrusters which can be failed to make sure that $rank(\mathbf{A}) = 6$.

Chapter 7

Dynamic Configuration-Umbrella Robot

Contents

A problem of robot's dynamic configuration is presented in this chapter. In particular, the robot adapts its configuration corresponding with desired control vector to minimize an objective function (an energy-like criterion). When desired control vector ($\mathbf{F}_B^d$) changes in each time step, an algorithm, called A-SQP, based on Sequential Quadratic Programming method, is proposed to solve the problem. Moreover, control allocation methods are investigated in this dynamic configuration problem.

7.1 Introduction

Nowadays, underwater researches have tremendously progressed because of new technologies, such as sensor techniques, electronic devices, machine learning algorithms. Many models of ROVs, AUVs have been developed in order to discover underwater environments [Yuh 2000][Zereik 2018], from under-actuated vehicles, i.e., torpedo shape for long-range missions, to over-actuated vehicles [Ropars 2018] [Dang 2019], i.e., symmetrical shape for station keeping or local environment observation. However, all these underwater robots have fixed configurations and their controllers have been designed to follow specified configurations. To increase the flexibility, configuration of underwater robots (URs) can be able to vary or be reconfigurable. A state of art for reconfigurable URs is presented in Chapter 6. Additionally, a ROV from SubseaTech [SubseaTech] can modify the actuator's angles but this mechanism is not shown clearly. Bio-inspired robots can be also considered as self-reconfigruable robots (as discussed in Chapter 1), i.e., snake robots [Transeth 2008]. To reduce the cost of building underwater vehicles (one robot can

carry out several tasks with different configurations) and to increase the versatility, a dynamically reconfigurable configuration of robots is needed and attractive.

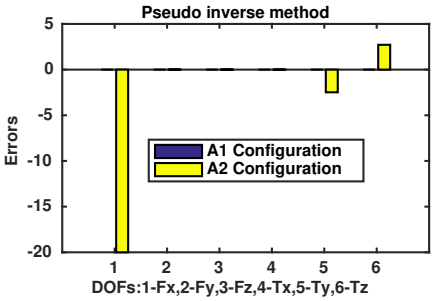
In general, the control diagram of an underwater robot is shown in Figure 4.1. The separation of control law and control allocation is useful to exploit advantages of actuator redundancy [Johansen 2013]. For dynamically reconfigurable robots, challenging questions arise and concern: control allocation, control law adaptation and modification of dynamic properties (e.g. modification of the meta-center). For simplicity, in this chapter, we assume that centers of mass and buoyancy are not changing during the mission (it can be achieved with a suitable design). The control system outputs a desired force vector, also called actuation demand ($\mathbf{F}_B^d$, expressed in the body-frame) which explicitly considers system dynamics and kinematic properties. It is then the role of control allocation to compute actuators inputs ($\mathbf{F}_m$) whose resulting action ($\mathbf{F}_B$) should realize the actuation demand ($\mathbf{F}_B = \mathbf{F}_B^d$). The relation between controller's output ($\mathbf{F}_B^d$) and the actuation inputs ($\mathbf{F}_m$) of a system is described by a mapping block, called control allocation (CA) (a state of art of CA methods is presented in Chapter 1). This mapping is normally described by a *configuration matrix*, denoted as $\mathbf{A}$. This chapter focuses on the question of optimal adaptation of the actuation configuration with respect to an energy-like criterion.

7.2 Dynamic Control Allocation-The singularities

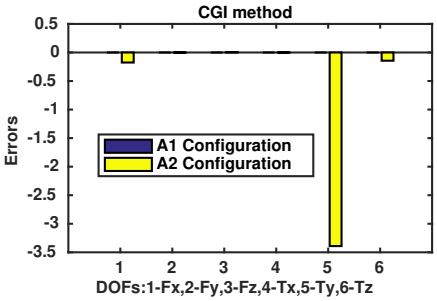
As aforementioned, there are many approaches to solve CA problem. However, in all these cases, a configuration matrix is constant and remains unchanged during robot's operation. For our robot, the configuration matrix is varying and properties of this matrix have to be studied carefully, because it yields to a singular configuration or nearly-singular configuration, i.e., the actuation system is redundant but results in an under-actuated system (some DOFs are not controllable). In the case of nearly-singularity, the minimum singular value of configuration matrix is too small. This yields that the pseudo-inverse is too big (it is easy to see with the SVD decomposition of the configuration matrix $\mathbf{A}$) and causes the big error if the pseudo-inverse based CA methods are used. One solution is that we can avoid the nearly-singular situation by neglecting too small singular values. However, this causes error and not suitable for dynamic configuration because configuration matrix can move from singular to nearly-singular configuration and it is difficult to determine conditions to cancel nearly-singularities.

We can verify this phenomena by an example of our robot. At two angles $\alpha_F = \alpha_B = 45^0$ (see Chapter 6), the robot's configuration matrix, $\mathbf{A}_1$, is singular and the robot can only go along X-axis (not Y and Z-axes), and at two angles $\alpha_F = \alpha_B = 45.02^0$ the robot's configuration matrix changes to $\mathbf{A}_2$, a nearly-singular matrix, and the robot is also controllable along X-axis easily. We investigate the errors of control allocation problem for these two configurations with different control allocation methods including pseudo inverse based (pure pseudo-inverse, cas-

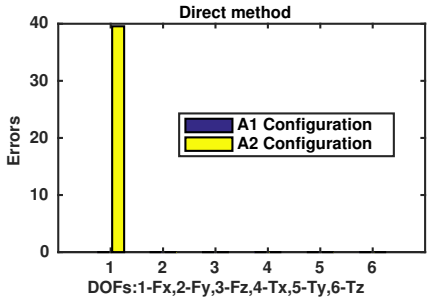
caded generalized inverse-cgi, direct method-dir) and nonlinear programming based approaches (sequential least square-sls, weighted least square-wls, minimal least square-mls, fixed point-fxp, interior point (ip) method) [Härkegård 2003].



(a) Errors of pure pseudo-inverse method



(b) Errors of cgi method



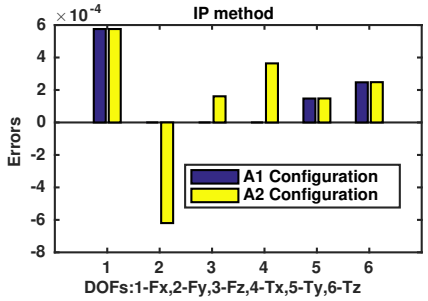
(c) Errors of dir method

Figure 7.1 – Errors of pseudo-based CA methods

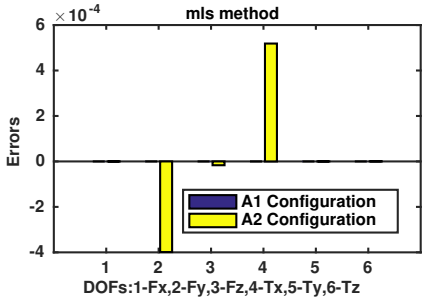
From Figure 7.1, when two angles α_F and α_B change a small value, the errors grow significantly for control allocation methods which based on pseudo-inverse for X-axis particularly (DOFs-1-Fx). The errors also remains for CA methods based on nonlinear programming (Figure 7.2), however, these values are quite small because these methods avoid nearly-singularity of configuration matrix by not using pseudo-inverse of matrix. It is easy to see that the nonlinear programming based control allocation methods are more suitable for the reconfigurable robot.

7.3 Dynamic configuration problem

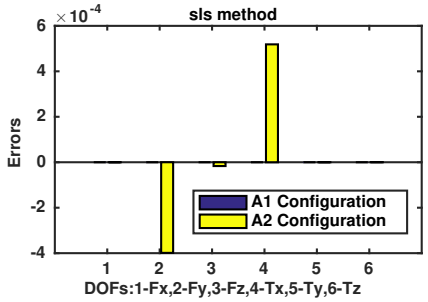
A prototype of dynamic configuration AUV, called Umbrella Robot, is presented in Chapter 6. In this section, we consider a dynamic configuration problem with respect to an energy-like criterion which is defined as the norm of actuation force vector, $\mathbf{F}_m$, applied on thrusters. This is reasonable thanks to the nearly-linear



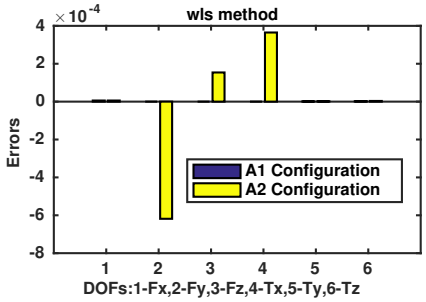
(a) Errors of ip method



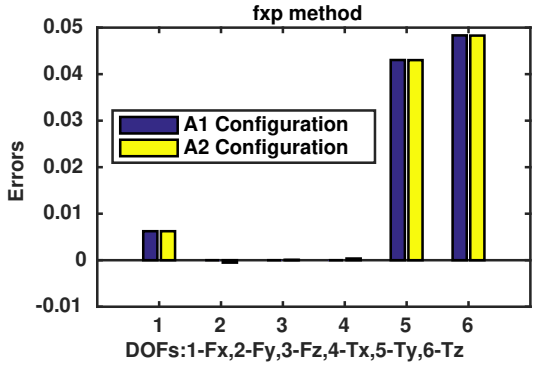
(b) Errors of mls method



(c) Errors of sls method



(d) Errors of wls method



(e) Errors of fxp method

Figure 7.2 – Errors of nonlinear programming based CA methods

characteristics of thrusters. The problem is formulated as:

$$\min_{\alpha_F,\alpha_B,\mathbf{F}_m} J = \|\mathbf{F}_m\|^2 \tag{7.1a}$$

$$s.t \quad 45^0 \leq \alpha_F, \alpha_B \leq 90^0 \tag{7.1b}$$

$$\mathbf{F}_m \in \mathbb{F} \tag{7.1c}$$

$$\mathbf{F}_B^d - \mathbf{A}(\alpha_F, \alpha_B)\mathbf{F}_m = 0 \tag{7.1d}$$

where $\mathbf{F}_B^d$ is desired force vector (an output from the controller) or actuation demand, $\mathbb{F}$ is a feasible set of thrusters forces. The constraint (7.1b) is mechanical limitations of the Umbrella Robot.

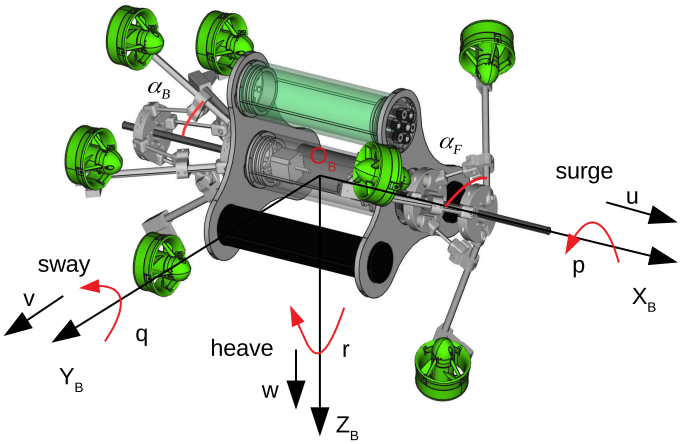


Figure 7.3 – Definitions of two angles α_F and α_B

The objective is to find two angles, α_F, α_B (see in Figure 7.3), and actuation force vector $\mathbf{F}_m$ in order to minimize an energy-like function J and satisfy constraints. This is a nonlinear optimization problem and is solved for each sampling time (online) because the desired force vector $\mathbf{F}_B^d$ is varied in each time step in general case. In our problem, the configuration matrix $\mathbf{A}$ is dynamic and belongs to two angles, α_F and α_B (elements of matrix $\mathbf{A}$ are detailed in Appendix A.3).

Other perspectives we have to consider are the reactivity of robot (the time for state propagation/system response) and time-delay of changing configurations. In fact, if the system response is too fast, we can not apply online optimization. Our objective is to solve online optimization problem, therefore, we assume that the time for solving one iteration of optimization problem is smaller the time that system propagates from current sates to next states. In our case, underwater robots, this assumption is reasonable.

For time-delay of changing configurations, at time step k, we have two angles α_{Fk} and α_{Bk}. At the next time step, $k+1$, assume that we get a solution from optimization problem with two angles $\alpha_{F(k+1)}$ and $\alpha_{B(k+1)}$. Physically, we have to spend time, $\triangle t_\alpha$, for changing from α_{Fk} to $\alpha_{F(k+1)}$ and from α_{Bk} to $\alpha_{B(k+1)}$, a new optimal configuration. If the changing time is too long, it is not associated to corresponding time step $k+1$. However, this changing time can not be too fast in virtue of limitations of DC motors. In particular, following Figure 7.4, at time step k, we have states of system, $\mathbf{x}_k$, two angles α_{Fk} and α_{Bk}, configuration matrix $\mathbf{A}_k$, PWM (Pulse Width Modulation) input values of thrusters, $\mathbf{c}_{mk}$, force vector applied on thrusters $\mathbf{F}_{mk}$, and resulting force vector in DOFs $\mathbf{F}_{Bk}$ (w.r.t body-frame)(suppose that $\mathbf{F}_{mk}$ and $\mathbf{F}_{Bk}$ can be measured or estimated). Starting the next time step, $k+1$, the desired control vector $\mathbf{F}_{B(k+1)}^d$ is received from output of the controller. The solution of dynamic configuration problem gives the two

desired angles of the step $\alpha^d_{F(k+1)}$ and $\alpha^d_{B(k+1)}$. From this point, mechanic system of Umbrella Robot (DC motors) will run to achieve these desired angles (with speed v). In the meantime, we also get the new desired configuration matrix $\mathbf{A}^d_{k+1}$, desired force vector applied on thrusters $\mathbf{F}^d_{m(k+1)}$, PWM vector $\mathbf{c}_{m(k+1)}$ which feeds to thrusters thanks to inverse thruster characteristic and we get $\mathbf{F}_{m(k+1)}$ thanks to thrusters actuation. It is clear that the changes of two angles must finish before this moment and we already achieved new configuration of the robot (and also new configuration matrix). After that, we have resulting vector in DOFs $\mathbf{F}_{B(k+1)}$. Otherwise, these two desired angles of time step $k+1$ will be achieved after several or many time steps later. This will cause undesired responses because of unassociated configuration. In order to solve our problem, two assumptions are described as follows:

1. Assumption 1: The reactivity of the system is long enough to solve one iteration of optimization problem, control allocation time, and basic operations.

2. Assumption 2: The time for changing mechanic system between two consecutive angles is fast enough in one sampling time.

In practice, the Assumption 1 is reasonable for Umbrella Robot. The Assumption 2 can be satisfied if the derivation between two consecutive angles is small enough.

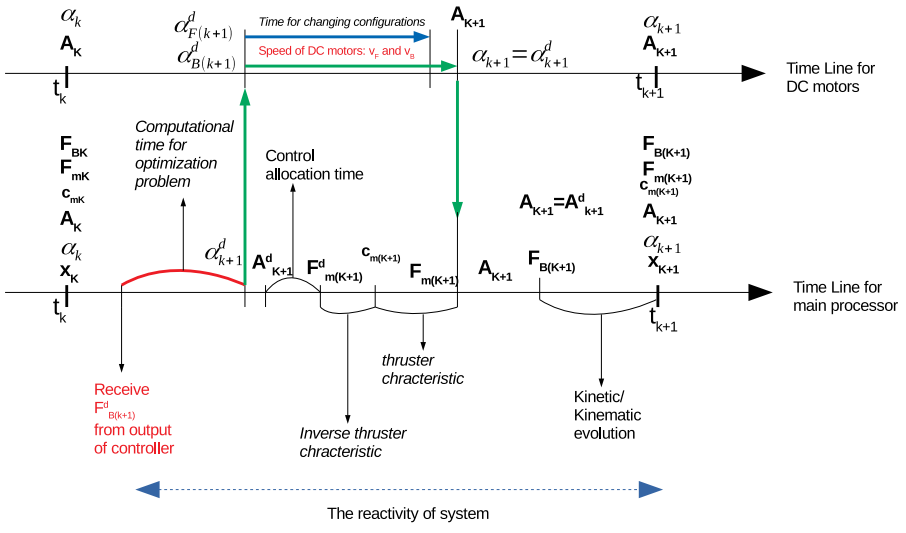


Figure 7.4 – Time line of main processor and DC motors

7.4 Problem solution

This section proposes an approach to solve online the problem (7.1). The constraint (7.1d) is strictly and not easy to solve, therefore we relax this constraint by adding it to the objective function with weights. This can rise the error of control allocation embedding in the problem. To enhance the performances, control allocation method is also used in one step of the main algorithm. This problem can be rewritten as:

$$\min_{\alpha_F,\alpha_B,\mathbf{F}_m} J = w_1\|\mathbf{F}_m\|^2 + w_2\|\mathbf{F}_B^d - \mathbf{A}(\alpha_F,\alpha_B)\mathbf{F}_m\|^2 \tag{7.2a}$$

$$s.t \quad 45^0 \leq \alpha_F, \alpha_B \leq 90^0 \tag{7.2b}$$

$$\mathbf{F}_m \in \mathbb{F} \tag{7.2c}$$

where w_1 and w_2 are scaling weights.

By denoting a vector $\mathbf{x} = [\alpha_F \quad \alpha_B \quad \mathbf{F}_m]^T$, the problem can be formulated as:

$$\min_{\mathbf{x}} f(\mathbf{x}) = w_1\|\mathbf{w}\mathbf{x}\|^2 + w_2\|(\overline{\mathbf{F}}_B^d - \overline{\mathbf{A}}(\mathbf{x})\mathbf{x})\|^2 \tag{7.3a}$$

$$s.t \quad \mathbf{x} \in \mathbb{X} \tag{7.3b}$$

where $\mathbb{X}$ is a box-constraint including limitations of two angles and saturations of thrusters, $\mathbf{w} = [0 \quad 0 \quad 1]^T$, $\overline{\mathbf{A}}(\mathbf{x}) = [\mathbf{0}^{n\times 1} \quad \mathbf{0}^{n\times 1} \quad \mathbf{A}]$, $\overline{\mathbf{F}}_B^d = [0 \quad 0 \quad (\mathbf{F}_B^d)^T]^T$.

The problem (7.3) can be rewritten in a compact form as:

$$\min_{\mathbf{x}} f(\mathbf{x}) \tag{7.4a}$$

$$s.t \quad \mathbf{c}_i^T\mathbf{x} \leq b_i, i = \overline{1...m} \tag{7.4b}$$

where $\mathbf{c}_i$ is a proper vector, m is the number of constraints.

In this part, we propose a A-SQP (Accelerating-Sequential Quadratic Programming) algorithm to solve online our problem. This is based on active-set SQP approach, an efficient method for small and medium nonlinear problem, with backgrounds presented in Chapter 3. We recall active-set definition hereafter.

Definition 15 *An active set of* (7.4) *is a set of constraints indices such that* $\mathbf{c}_i^T\mathbf{x} = b_i$. *Specifically,* $\mathcal{A} = \{i|\mathbf{c}_i^T\mathbf{x} = b_i\}$

Suppose the active set of (7.4) is given, we consider our problem only with equality constraints as:

$$\min_{\mathbf{x}} f(\mathbf{x}) \tag{7.5a}$$

$$s.t \quad \mathbf{c}_i^T\mathbf{x} = b_i, i \in \mathcal{A} \tag{7.5b}$$

The Lagrangian function of (7.5) is $L(\mathbf{x},\boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{i\in\mathcal{A}}\lambda_i(\mathbf{c}_i^T\mathbf{x} - b_i)$. The optimal KKT (Karush-Kuhn-Tucker) condition of this problem at local optimal point $(\mathbf{x},\boldsymbol{\lambda})$ can be written as:

$$\nabla_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\lambda}) = \nabla f(\mathbf{x}) + \sum_{i \in \mathcal{A}} \lambda_i \mathbf{c}_i = \mathbf{0} \tag{7.6a}$$

$$\mathbf{c}_i^T \mathbf{x} = b_i, i \in \mathcal{A} \tag{7.6b}$$

The problem (7.6) is rewritten as a compact form:

$$F(\mathbf{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \nabla f(\mathbf{x}) + \mathbf{C}^T \boldsymbol{\lambda} \\ \mathbf{C}\mathbf{x} - \mathbf{b} \end{bmatrix} = \mathbf{0} \tag{7.7}$$

where $\mathbf{C} = [\mathbf{c}_i^T], \quad \boldsymbol{\lambda} = [\lambda_i] \quad \mathbf{b} = [b_i], \quad i \in \mathcal{A}$.

One approach to solve nonlinear equations (7.7) by using Newton's method. The Jacobian of (7.7) with respect to $(\mathbf{x}, \boldsymbol{\lambda})$ is given by:

$$F'(\mathbf{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \nabla^2 L(\mathbf{x}) & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \tag{7.8}$$

The direction for the next step of problem (7.5) is a solution of (7.7) and is computed as:

$$\begin{bmatrix} \nabla^2 L(\mathbf{x}) & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{p}_{x_k} \\ \mathbf{p}_{\lambda_k} \end{bmatrix} = \begin{bmatrix} -\nabla f(\mathbf{x}) - \mathbf{C}^T \boldsymbol{\lambda} \\ -\mathbf{C}\mathbf{x} + \mathbf{b} \end{bmatrix} \tag{7.9}$$

At each iteration of solver, denoted as step k, we have an active set , called a working set $\mathcal{W}_k$. One constraint can be added or eliminated through the working set. This can be done by checking the sign of Lagrange multipliers $\lambda_i \geq 0$ thanks to KKT conditions of (7.4). If we correctly identify the optimal active set then our problem will converge rapidly by Newton's method as the aforementioned analysis. However, optimal active set is not easy to determine.

Thanks to the efficient and popular of convex quadratic programming which can converge in milliseconds [Ferreau 2014], at step k with $\mathbf{x}_k$, we consider another quadratic problem:

$$\min_{\mathbf{p}} \frac{1}{2} \mathbf{p}^T \nabla^2_{\mathbf{xx}} L \mathbf{p} + \nabla f^T \mathbf{p} \tag{7.10a}$$

$$s.t \quad \mathbf{C}\mathbf{p} = -\mathbf{C}\mathbf{x}_k + \mathbf{b} \tag{7.10b}$$

The optimality conditions of (7.10) with Lagrange multipliers $\boldsymbol{\lambda}_q$:

$$\nabla^2_{\mathbf{xx}} L \mathbf{p} + \nabla f + \mathbf{C}^T \boldsymbol{\lambda}_{\boldsymbol{q}} = \mathbf{0} \tag{7.11a}$$

$$\mathbf{C}\mathbf{p} - \mathbf{b} + \mathbf{C}\mathbf{x}_k = \mathbf{0} \tag{7.11b}$$

This is equivalent with:

$$\nabla^2_{\mathbf{xx}} L \mathbf{p} + \nabla f + \mathbf{C}^T \boldsymbol{\lambda}_{\boldsymbol{q}} - \mathbf{C}^T \boldsymbol{\lambda} = -\mathbf{C}^T \boldsymbol{\lambda} \tag{7.12a}$$

$$\mathbf{C}\mathbf{p} - \mathbf{b} + \mathbf{C}\mathbf{x}_k = \mathbf{0} \tag{7.12b}$$

or:

$$\begin{bmatrix} \nabla^2 L & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ \boldsymbol{\lambda_q} - \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\nabla f - \mathbf{C}^T \boldsymbol{\lambda} \\ -\mathbf{C}\mathbf{x}_k + \mathbf{b} \end{bmatrix} \tag{7.13}$$

It is easy to see that (7.9) and (7.13) are almost the same except that:

$$\mathbf{p}_{\lambda_k} = \boldsymbol{\lambda_q} - \boldsymbol{\lambda} \tag{7.14}$$

In order to find the direction $[\mathbf{p}_x \quad \mathbf{p}_\lambda]^T$, instead of solving (7.9), we solve quadratic problem (7.10) and update Lagrange multipliers as (7.14). Moreover, Hessian matrix $\nabla^2 L$ is approximated by BFGS (Broyden-Fletcher-Goldfarb-Shanno) formula to make sure it is positive definite or semi-definite and problem (7.10) is quadratic convex problem which always exits a global optimal solution.

The idea is extended to the problem with inequality (7.4) by solving the sub-quadratic programming as follows to find the direction:

$$\min_{\mathbf{p}} \frac{1}{2} \mathbf{p}^T \nabla^2_{\mathbf{xx}} L \mathbf{p} + \nabla f^T \mathbf{p} \tag{7.15a}$$

$$s.t \quad \mathbf{C}\mathbf{p} \leq -\mathbf{C}\mathbf{x}_k + \mathbf{b} \tag{7.15b}$$

This is possible because the set of active constraints $\mathcal{A}_k$ at the solution of (7.15) constitutes the guess of the active set at the solution of the nonlinear program (7.4) [Theorem 18.1 in [Nocedal 2006]]. Algorithm 4 shows our procedure to solve online dynamic configuration.

Algorithm 4 A-SQP optimal configuration algorithm

Input: desired control inputs $\mathbf{F}_B^d$ (from controller)
Output: Optimal angles α_F, α_B and thruster forces $\mathbf{F}_m$

1: Initialization: primal-dual parameters $(\mathbf{x}_0, \boldsymbol{\lambda}_0)$, Hessian approximation $\mathbf{B}_0$
2: **for** $k = 1 <= 2$ **do**
3: Evaluate f, ∇f at $\mathbf{x}_k$
4: Compute direction $\mathbf{p}_k$ by solving (7.15) and have corresponding Lagrange multipliers $\boldsymbol{\lambda}_{\boldsymbol{q_k}}$
5: Compute direction $\mathbf{p}_{\lambda_k} = \boldsymbol{\lambda}_{\boldsymbol{q_k}} - \boldsymbol{\lambda}$
6: Choose step length $\alpha_k = 1$ (maximize the direction)
7: Update $\mathbf{x}_k = \mathbf{x}_k + \alpha \mathbf{p}_k$ and $\boldsymbol{\lambda_k} = \boldsymbol{\lambda_k} + \alpha \mathbf{p}_{\lambda_k}$
8: Set $\mathbf{s}_k \leftarrow \alpha_k \mathbf{p}_k$ and $\mathbf{y}_k \leftarrow \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}_{k+1}, \boldsymbol{\lambda}_{k+1}) - \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}_k, \boldsymbol{\lambda}_{k+1})$
9: Update $\mathbf{B}_{k+1}$ using BFGS formula (see Algorithm 2 in Chapter 3)
10: **end for**

7.5 Control Design for a dynamic configuration system

In this part, we investigate effects of dynamic configuration for controller design. In fact, our system can vary from under-actuated to over-actuated one, the

corresponding controllers are designed differently. Normally, model-based controller design belongs a chosen model, however, when the model is changed, the controller has to be redesigned. For a dynamic configuration robot, two controllers are ready during its operation: one for under-actuated configuration and one for fully or over-actuated configuration (the difference between fully and over-actuated system is control allocation approach). One proper mechanism will be designed to switch between controllers. For examples, a unified function was derived in [Breivik 2006] and a switched function was proposed in [Xiang 2015] which belongs to the velocity of robot as Equation (7.16).

$$F_i = f(\boldsymbol{\nu})F_{u,i} + (1 - f(\boldsymbol{\nu}))F_{f,i} \tag{7.16}$$

where $F_{u,i}$ and $F_{f,i}$ are resulting forces corresponding with under-actuated and fully-actuated systems respectively, $i \in \{u, v, w, p, q, r\}$ representing the AUV DOFs, $f(\boldsymbol{\nu})$ is a function of AUV's velocity and is chosen to guarantee the smooth transition between configurations, i.e., sinusoid or tanh function.

In our work, the output of controller ($\mathbf{F}_B^d$) is the input of the problem (7.1) and a solution of this problem is two angles α_F and α_B. The properties of system belongs to these two angles (under/fully/or over-actuated system). So we do not need a switch mechanism here, and just one controller in 6 DOFs and robot will adjust configuration with corresponding two angles. However, one challenging question arises here: it should be have a guidance law of $\mathbf{F}_B^d$ for our problem. This relates to multi-parametric programming and out of scope of this work and will be a future studies. In the following simulations, we investigate A-SQP with different kinds of $\mathbf{F}_B^d$.

7.6 Conclusion

This chapter presented a dynamic configuration problem of a robot which can modify its actuation configurations during its missions. Some questions derives for this such as dynamic control allocation problem and optimal configuration for a mission. An investigation of dynamic control allocation was presented and we can see that nonlinear programming based CA methods are suitable for dynamic configuration case because of nearly-singularity issue. An optimal problem w.r.t energy-like criterion of the robot was suggested and an algorithm, called A-SQP based on Sequential Quadratic Programming with active-set method, was proposed. The algorithm is carried out in each time step. Note that, for dynamic configuration, the robot can change from an under-actuated system to over-actuated system. The controller design for this change was discussed and not considered in this thesis.

Chapter 8

Reconfigurable and Dynamic Configuration: Simulation and Experiment Results

Contents

In this chapter, simulation and experiment results for Umbrella Robot are shown. For reconfigurable capability, manipulability index and force/torque spaces are simulated to illustrate the varying of these performances when changing robot's configuration. In case of dynamic configuration problem, simulation results with different issues, i.e., given missions, path following, and observation case, are studied to show efficiency of proposed method. Experiments with basic missions and an integrated one are tested on the current prototype of the Umbrella Robot (UR). The results show the robot's versatility.

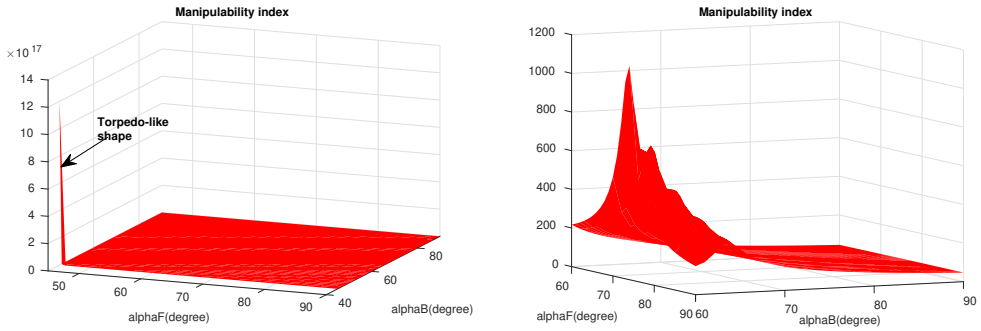
8.1 Simulations

8.1.1 Reconfigurable configuration

The simulation results display the properties of system when modifying the configuration of system. These are attainable force/torque spaces and *manipulability index* [Dang 2019] when changing the **A** matrix. The thrusters characteristics are from Blue Robotics as shown in Fig 5.7. When the values of these angles go from 45° to 90°, the manipulability index of UR is drawn in Figure 8.1(a). The values of this index in some configurations are shown in Table 8.1. The minimum value is achieved at $\alpha_F = \alpha_B = 90°$ and maximum value is at $\alpha_F = \alpha_B = 45°$ (the UmRobot is like a torpedo-shape robot).

Angles	Manipulability index	Value
$\alpha_F = \alpha_B = 45°$	I_m	∞
$\alpha_F = 50°, \alpha_B = 60°$	I_m	157.2609
$\alpha_F = \alpha_B = 90°$	I_m	48.7762

Table 8.1 – Manipulability index with different configurations

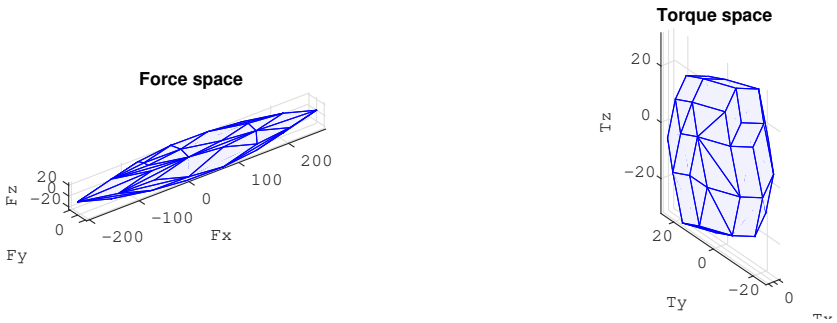


(a) Manipulability index of Umbrella robot w.r.t two angles α_F and $\alpha_B \in (45° \div 90°)$

(b) Manipulability index of Umbrella robot w.r.t two angles α_F and $\alpha_B \in (60° \div 90°)$

Figure 8.1 – Manipulability index of Umbrella robot w.r.t two angles

The attainable spaces (force and torque spaces) with two different configurations are shown in Figure 8.2 and Figure 8.3 (the units of forces is N and torques is N.m). It is obvious to see that attainable force space with $\alpha_F = \alpha_B = 90°$ (Figure 8.2(a)) is more isotropic than one with $\alpha_F = 50°, \alpha_B = 60°$ (Figure 8.3(a)).



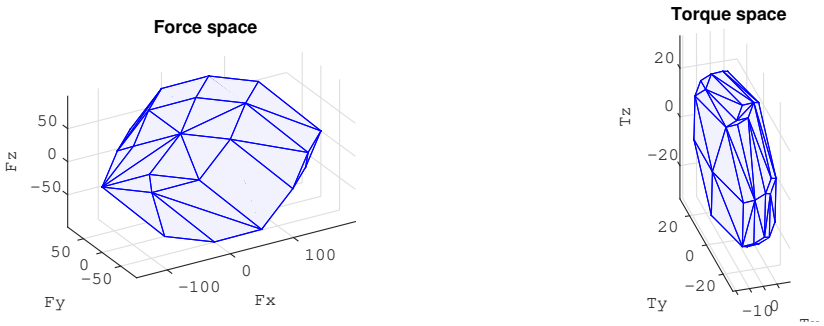
(a) Attainable force space

(b) Attainable torque space

Figure 8.2 – Attainable space ((a)-Force space, (b)-Torque space) with $\alpha_F = 50°, \alpha_B = 60°$

8.1.2 Dynamic configuration

For a dynamically configurable robot, various strategies, which depend on the mission requirements, can be applied because our robot can vary from an under-



(a) Attainable force space

(b) Attainable torque space

Figure 8.3 – Attainable space ((a)-Force space, (b)-Torque space) with $\alpha_F = \alpha_B = 90°$

actuated system to an over-actuated one and vice-versa. For simulations, dynamic parameters of robot are approximated by numerical software, i.e, ANSYS (see Appendix A.4). A simulated robot has been built and is shown in Figure 8.4. Note that in the simulations, external disturbances and model uncertainties are not taken into account. We know that varying configuration of robot belongs to desired force

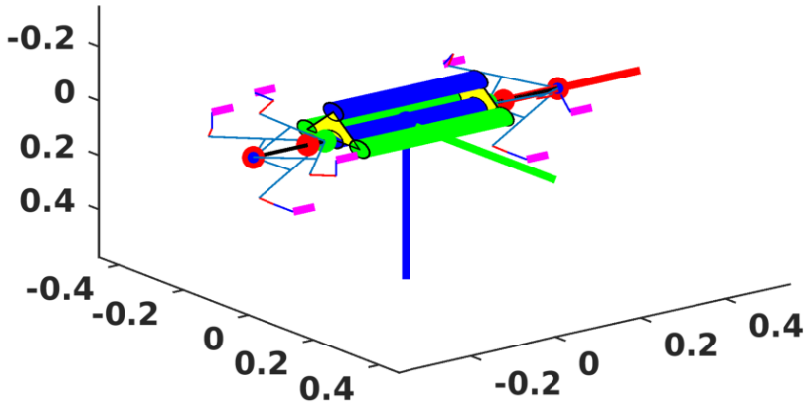


Figure 8.4 – Simulated robot

vector, $\mathbf{F}_B^d$, that can be statically given (in case where we know clearly desired actions on each DoF) or dynamically from output of a controller. We simulate and investigate the umbrella robot with two cases. In other side, if two angles α_F and α_B are chosen before a mission, the robot can operate as under-actuated system or over-actuated one. In this situation, it should have two controllers for each kind of systems and a commutation mechanism is needed. This is not studied in this work and will be a future work as aforementioned.

8.1.2.1 Static desired force vector (given)

In this part, a given mission consists of three stages which are: *i)* the robot goes straight along X-axis, *ii)* after that goes along X-axis and Y-axis, *iii)* and finally dives along Z-axis. The desired force vector, $\mathbf{F}_B^d$, of this mission is given as: $\mathbf{F}_B^d = [10 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$, $\mathbf{F}_B^d = [10 \quad 10 \quad 0 \quad 0 \quad 0 \quad 0]^T$, and $\mathbf{F}_B^d = [0 \quad 0 \quad 10 \quad 0 \quad 0 \quad 0]^T$ respectively. Note that this force vector is applied on body-frame, so X-Y-Z axes are of body-frame. We compare three following cases as:

1. The robot carries out the mission with a fixed-configuration with arbitrary angles (for example, two angles $\alpha_F = 90, \alpha_B = 90$, and $\alpha_F = 60, \alpha_B = 70$)

2. The robot carries out the mission with optimal fixed-configurations. It means that, for each stage of the mission, optimal angles which can be found by applying Global search methods (see Table 8.2).

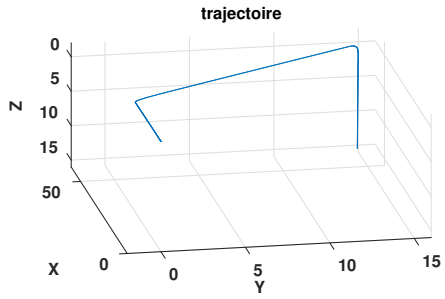
3. The robot carries out the mission with dynamic configuration (A-SQP algorithm).

Desired force	Optimal Angles(Degree)
$\mathbf{F}_B^d = [10 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$	$\alpha_F = 45, \alpha_B = 45$
$\mathbf{F}_B^d = [10 \quad 10 \quad 0 \quad 0 \quad 0 \quad 0]^T$	$\alpha_F = 45, \alpha_B = 90$
$\mathbf{F}_B^d = [0 \quad 0 \quad 10 \quad 0 \quad 0 \quad 0]^T$	$\alpha_F = 45, \alpha_B = 90$

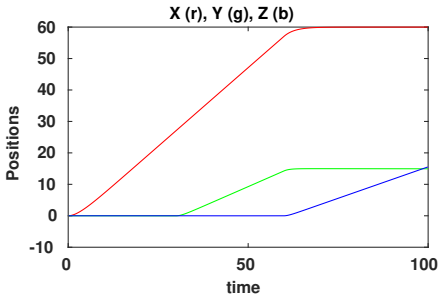
Table 8.2 – Corresponding optimal angles of desired force

Note that in case of varying configurations during the mission, the time response of DC motors is taken into account. It means that the velocity of DC motors satisfies Assumption 2.

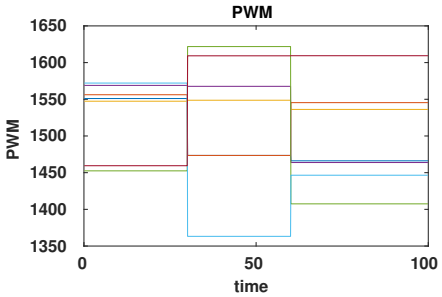
Case 1: In this case, the robot carries out a mission with fixed-configuration. The simulation results of robot's trajectory, robot's positions, PWM inputs of thrusters are shown in Figure 8.5 with $\alpha_F = 90^0$, $\alpha_B = 90^0$ and Figure 8.6 with $\alpha_F = 60^0$, $\alpha_B = 70^0$.



(a) Robot's trajectory

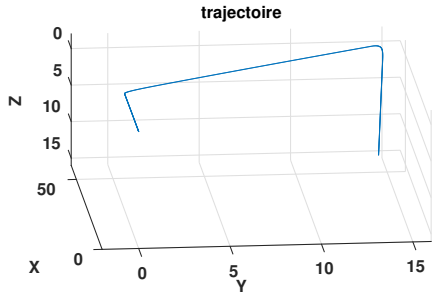


(b) Robot's positions

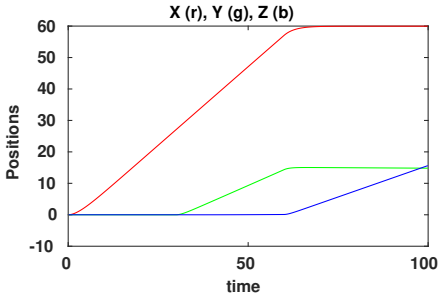


(c) PWM evolution

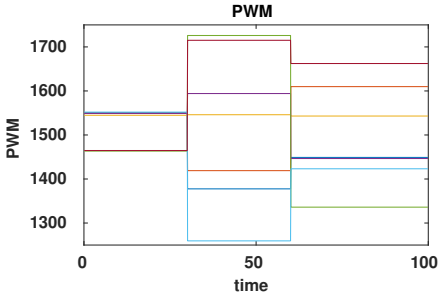
Figure 8.5 – Fixed-configuration simulation results (desired command vector is given) ($\alpha_F = 90^0$, $\alpha_B = 90^0$)



(a) Robot's trajectory



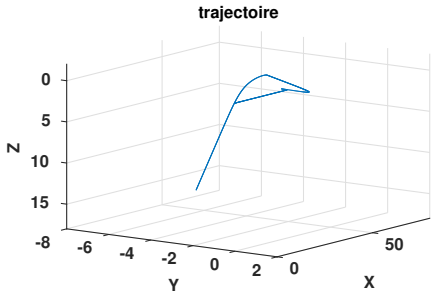
(b) Robot's positions



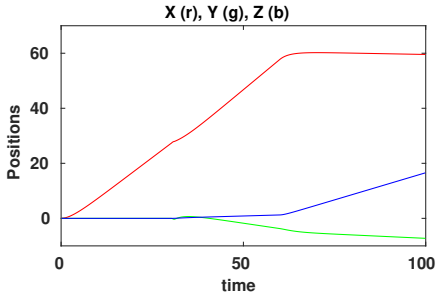
(c) PWM evolution

Figure 8.6 – Fixed-configuration simulation results (desired command vector is given) ($\alpha_F = 60^0$, $\alpha_B = 70^0$)

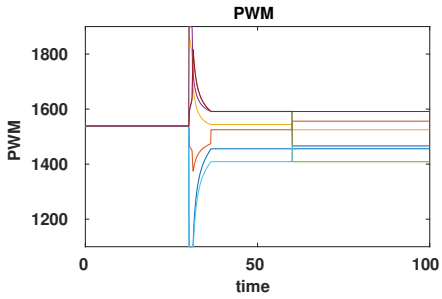
Case 2: The robot carries out a mission with different optimal fixed-configurations. The simulation results of robot's trajectory, robot's position, and PWM inputs of thrusters are shown in Figure 8.7. In this case, for transition time between two consecutive optimal fixed-configurations, robot got fluctuation.



(a) Robot's trajectory



(b) Robot's positions

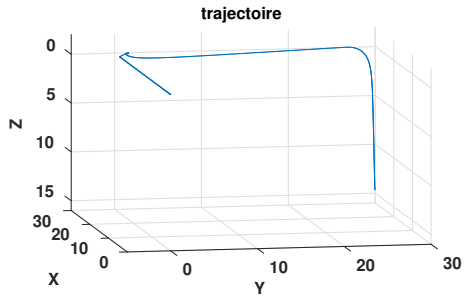


(c) PWM evolution

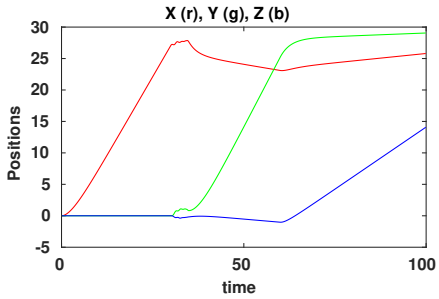
Figure 8.7 – Optimal fixed-configuration simulation results (desired command vector is given)

Case 3: In this case, the robot carries out a mission with dynamic configuration. The simulation results of robot's trajectory, robot'positions, PWM inputs of thrusters, and evolution of two angles are shown in Figure 8.8.

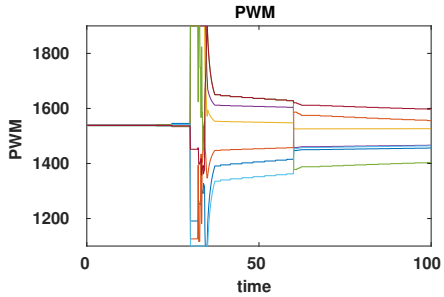
We compare three cases with respect to an energy-like criterion which is calculated as norm of actuation vector, $\|\mathbf{F}_m\|^2$. The evolution of energy-like criterion is presented in Figure 8.9. It is obvious to see that energy-like criterion of optimal fixed-configuration is lowest in each stage of mission. For dynamic configuration, it goes lower after each time step except in a transition period. This is also happen with optimal fixed-configuration. This proves the efficiency of our algorithm to minimize the energy of robot's operation. However, in transition time, for optimal-fixed configuration, the energy-like index is higher because robot spent time to vary between two consecutive optimal configurations. For transition time of dynamic configuration, the derivation between two consecutive angles is small enough, and some of DOFs are not completely controllable. Therefore, some thrusters reach saturation and energy-like index is quite high.



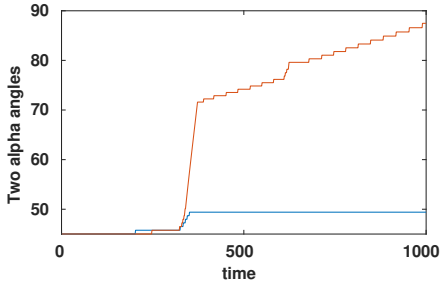
(a) Robot's trajectory



(b) Robot's positions



(c) PWM evolution



(d) Two angles evolution

Figure 8.8 – Dynamic configuration simulation results (desired command vector is given)

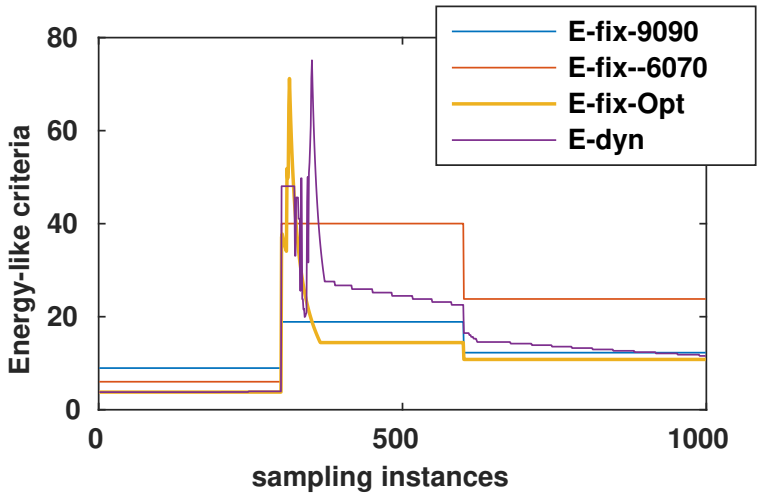


Figure 8.9 – Evolution of energy-like criterion with different cases

8.1.2.2 Dynamic desired force vector (output of a controller)

In general, a robot operates with control loop in which controller derives a desired force vector $\mathbf{F}_B^d$. In this section, we investigate the problem (7.1) when $\mathbf{F}_B^d$ value is dynamic. We simulate with two missions: path following and observation

problems which are very important in underwater robotics. Note that, in this case, desired vector force, $\mathbf{F}_B^d$, can be varied and be different in each time step.

Path-following problem For path following problem, readers can see methods in Appendix A.5. A Line-of-Sight (LoS) based guidance method is used in this simulation. We do a comparison of energy-like criterion between a static configuration and dynamic one with this mission. A chosen path is a spatial ellipse which is parameterized as follows:

$$x = 60\cos(0.2618t) \tag{8.1}$$
$$y = 60\sin(0.2618t) \tag{8.2}$$
$$z = \sin(0.2618t) + 5 \tag{8.3}$$

where t is a path parameter.

The desired composite speed $\mathbf{U}_d = 2m/s$. The initial posture of AUV is $[x(0) \quad y(0) \quad z(0) \quad \phi(0) \quad \theta(0) \quad \psi(0)]^T = [64 \quad 3 \quad 0 \quad 0 \quad 0 \quad 3\pi/4]^T$. The initial speed of AUV is $[u(0) \quad v(0) \quad w(0) \quad p(0) \quad q(0) \quad r(0)]^T = [1.5 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$.

The controller is designed as in Equations (A.112) to (A.115), we simulate the following cases:

Case 1: The robot is controlled to follow the predefined path with chosen two angles and the previous controller. In particular, $\alpha_F = \alpha_B = 70^0$ are chosen.

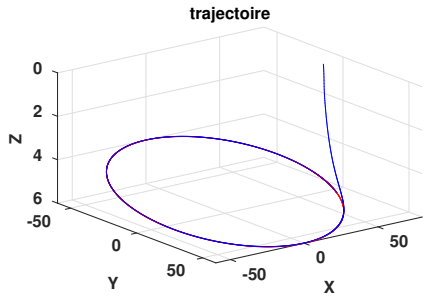
Case 2: Like the same with *Case* 1, the robot is controlled to follow the predefined path with chosen two angles and the previous controller controller. However, $\alpha_F = \alpha_B = 90^0$ are chosen because we want to investigate different fixed-configurations for path following problem.

Case 3: The robot is controlled to follow the predefined path with dynamic configuration.

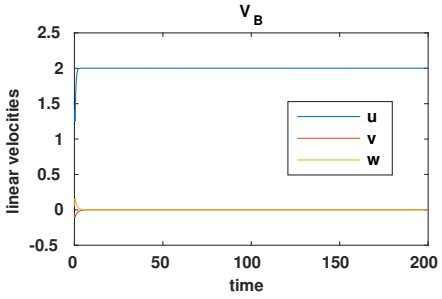
In order to evaluate the efficient of our approach with dynamic configuration, an energy-like criterion is compared with three cases. The next paragraphs show simulation results of three cases.

Case 1: Path following for $\alpha_F = \alpha_B = 70^0$: The robot is considered as an over-actuated system. A controller for under-actuated system does not guarantee the performances of path following problem. The PID controller considering the effect of roll angle (Equation A.115) is used. Simulation results for this case with trajectory, linear velocities, and PWM of thrusters are presented as in Figure 8.10.

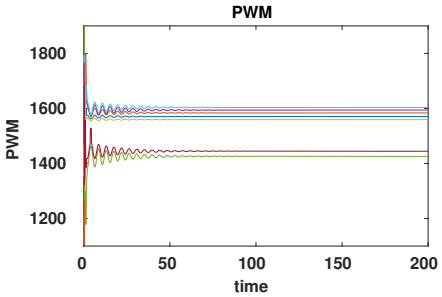
Case 2: Path following for $\alpha_F = \alpha_B = 90^0$. The simulation results are shown in Figure 8.11



(a) trajectory of Robot

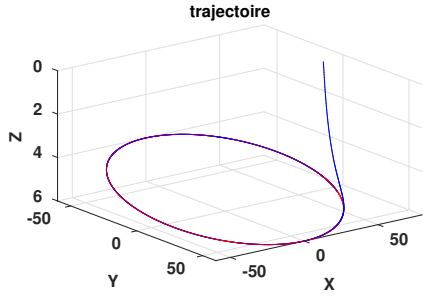


(b) Linear velocities

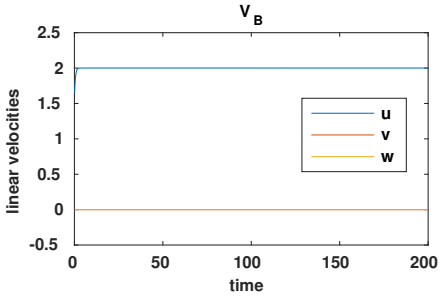


(c) PWM of thrusters

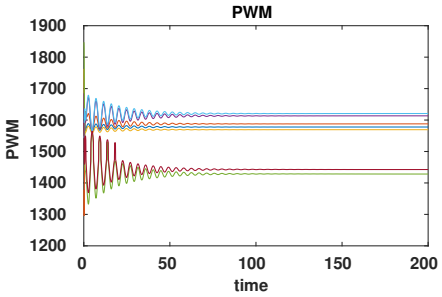
Figure 8.10 – Path following for ellipse with over-actuated configuration ($\alpha_F = \alpha_B = 70^0$)



(a) trajectory of Robot



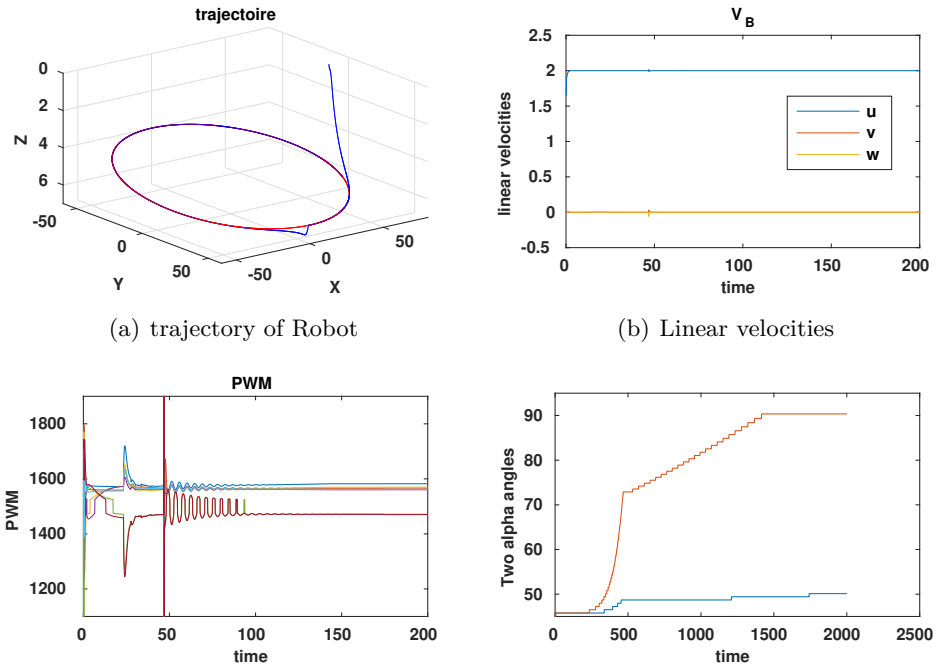
(b) Linear velocities



(c) PWM of thrusters

Figure 8.11 – Path following for ellipse with over-actuated configuration ($\alpha_F = \alpha_B = 90^0$)

Case 3 : Controller for dynamic configuration: In this case, the robot follows a path with dynamic configuration. It means that two angles (α_F and α_B) vary to associated desired vector force $\mathbf{F}_B^d$. Simulation results are shown in Figure 8.12



(a) trajectory of Robot

(b) Linear velocities

(c) PWM of thrusters

(d) evolution of two angles

Figure 8.12 – Path following for ellipse with dynamic configuration (full/over-actuated controller)

Energy-like criterion evolution for path following problem is shown in Figure 8.13.

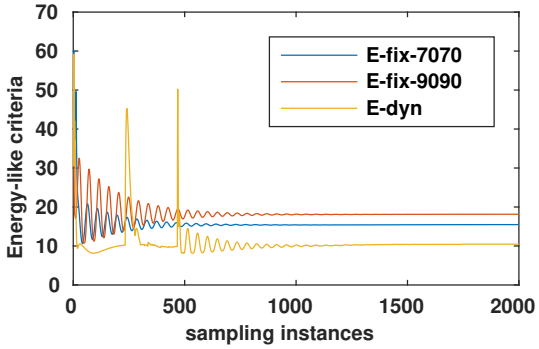


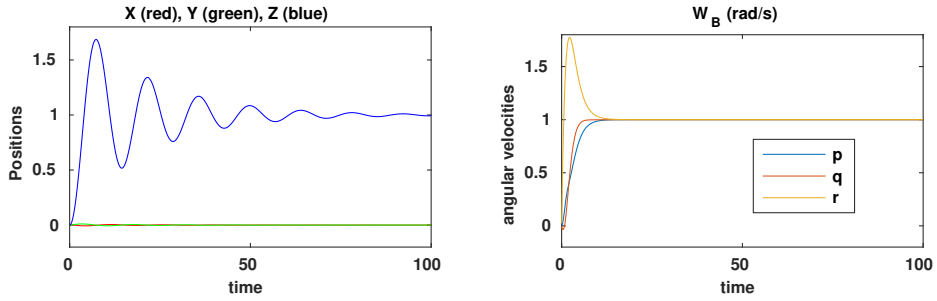
Figure 8.13 – Energy-like criteria for Path following problem

It is clear to see that the path-following performance is almost guaranteed for all

three cases. However, for dynamic configuration, robot's configuration (**A** matrix) goes through bad points in which performance of control allocation is not satisfied. This can be seen in Figure 8.12(a) and Figure 8.12. It will be an interesting future work. On energy perspective, dynamic configuration shows better than two others except two bad points which cause saturation of thrusters.

Observation problem

For observation problem, robot has normally to rotate some DoFs. This can not be carried out by under-actuated system which has some uncontrollable DoFs. Thanks to its versatility, our dynamic configuration can perform this mission easily. In this part, we present the simulation results of observation problem with the Umbrella robot in which the robot dives to constant depth with desired angular velocities, i.e., $\boldsymbol{\eta_{1des}}(m) = [x \quad y \quad z]^T = [0 \quad 0 \quad 1]^T$ and $\boldsymbol{\nu_{2des}}(rad/s) = [p \quad q \quad r]^T = [1 \quad 1 \quad 1]^T$. The model of simulation robot is shown in Figure 8.4. The controller is designed with quaternion techniques[Louis 2017] (see Appendix A.2 for quaternion backgrounds). The simulations include fixed and dynamic configurations. For dynamic case, we compare our algorithm with *Fmincon* function in Matlab Toolbox. The simulation results of fixed configurations are shown in Figures 8.14(a) 8.14(b)) in which $\alpha_F = \alpha_B = 90^0$. The simulation results of dynamic configurations are depicted in Figure 8.15 and 8.16. Note that in simulation, we assume that all states of robot can be measured/estimated completely. Moreover, warm start technique is used to accelerate the computational time of ASQP algorithm.

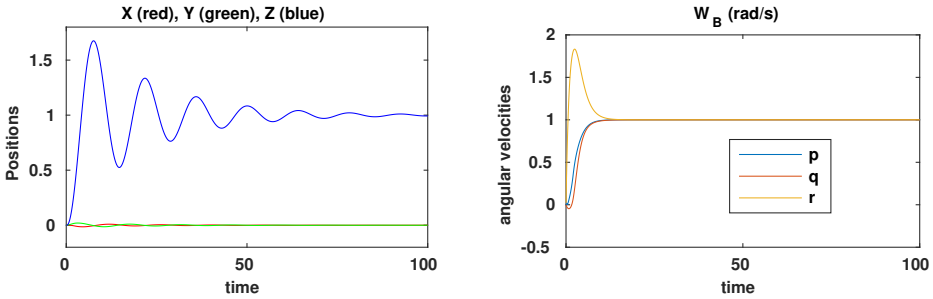


(a) Positions of Robot (b) Angular velocities

Figure 8.14 – Simulation results with fixed configurations

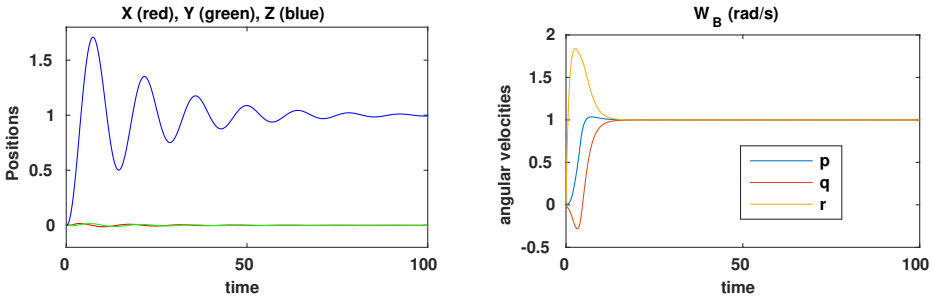
It is obvious to see that the control performances are guaranteed in static and dynamic configuration. However, the energy-like criterion is different. Specifically, the energy-like criteria of simulations cases which guarantee the control performances is illustrated in Figure 8.17(a). The dynamic configuration cases (with *Fmincon* and A-SQP) outperform static configuration one.

Our algorithm shows the same performances in comparison with *Fmincon* function. A comparison of computational time between methods including A-SQP, Fmincon at each sampling time is shown in Figure 8.17(b). Note that the computational time is not the same at each sampling time because of the properties of numerical accuracy. With the same energy-like criterion, we can see that A-SQP



(a) Positions of Robot (b) Angular velocities

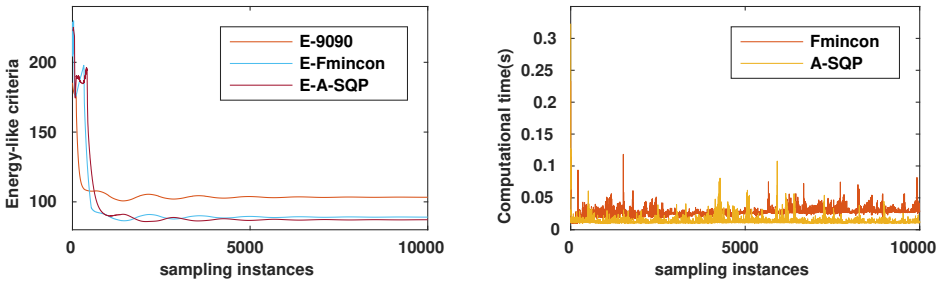
Figure 8.15 – Simulation results with dynamic configuration (Fmincon)



(a) Positions of Robot (b) Angular velocities

Figure 8.16 – Simulation results with dynamic configuration (A-SQP)

outperforms in computational time and suitable for a real test.



(a) Energy-like criterion evolution (b) Computationa time

Figure 8.17 – Energy-like criterion and computational time comparison

Considering the trade-off between the energy criteria and computational time, our approach can be acceptable for the versatility of the robot.

8.2 Experiments

The Umbrella robot is tested at a swimming pool (see Figure 8.18) with basic missions, such as yaw, depth, and surge control, and a complex mission.

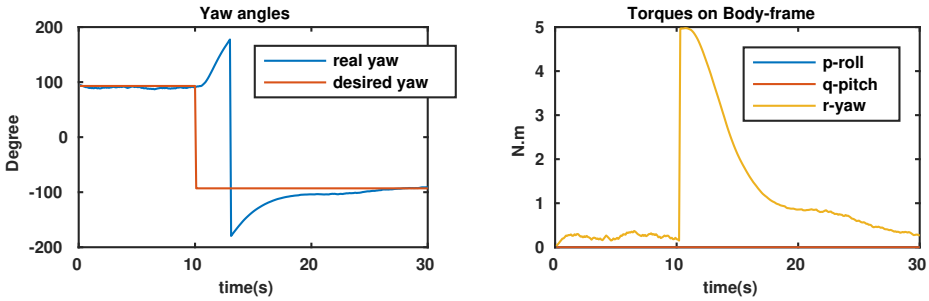
Figure 8.18 – Umbrella Robot at the swimming pool

8.2.1 Basic missions

This section presents basic experiments of the UmRobot such as surge control, yaw and depth control. PID/backstepping method with quaternion formalism was implemented.

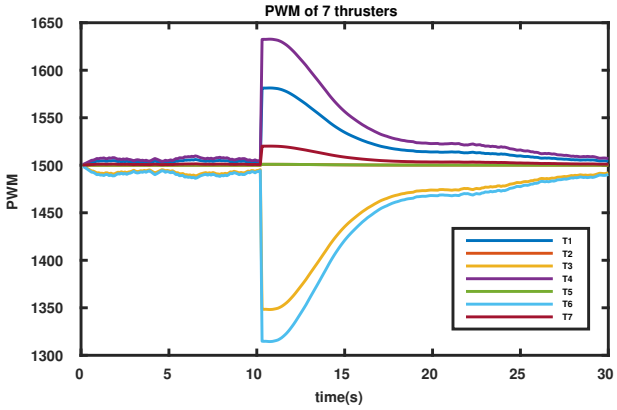
8.2.1.1 Yaw control

In this experiment, the desired yaw angle is set 93^0 (initial yaw angle of the robot) and -93^0. The UmRobot maintains initial yaw angle and make a turn to second desired angle (-93^0). In this test, robot's configuration is chosen as $\alpha_F = \alpha_B = 45^0$ (as a Torpedo shape). The experiment results are shown in Figure 8.19. In particular, yaw angle of robot follows desired yaw angle as in Figure 8.19(a). Applied torques and PWM inputs of thrusters are presented in Figure 8.19(b) and Figure 8.19(c) respectively.



(a) Yaw angles

(b) Applied torques



(c) PWM of thrusters

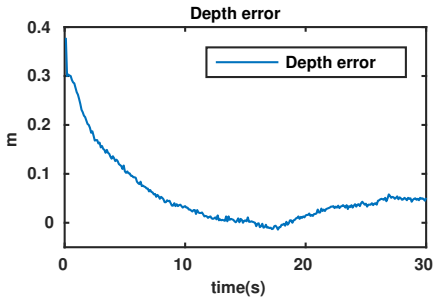
Figure 8.19 – Yaw control

8.2.1.2 Depth control

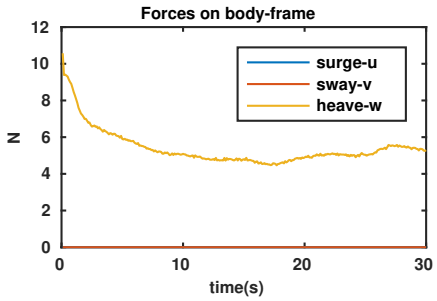
In this test, robot's configuration is set as $\alpha_F = \alpha_B = 70^0$ and the desired depth is a constant. The experiment results are shown in Figure 8.20. The depth error keeps stable as in Figure 8.20(a).

8.2.1.3 Surge-Pitch-Yaw control

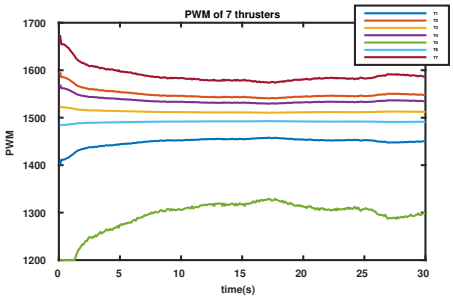
In this experiment, surge control is applied with $F_u = 25N$ and robot's direction is maintained (pitch and yaw control). The robot's configuration is $\alpha_F = \alpha_B = 45^0$. The experiment results are shown in Figure 8.21. Readers can see more with attached video links. It is obvious to see that Euler angles (roll, pitch, and yaw) hold stable during the mission (Figure 8.21(a)).



(a) Depth error

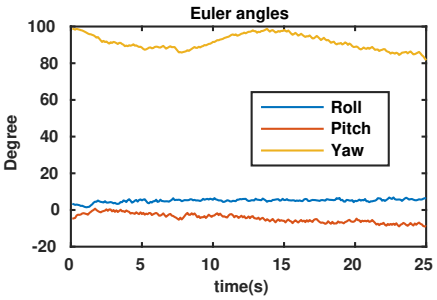


(b) Applied forces

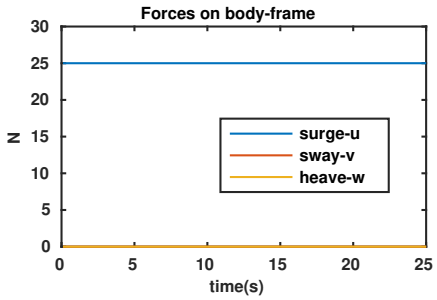


(c) PWM of thrusters

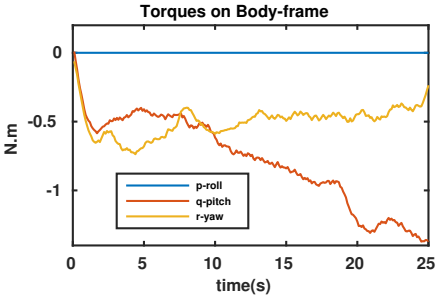
Figure 8.20 – Depth control



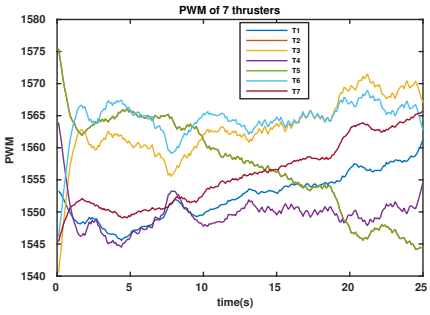
(a) Euler angles



(b) Applied forces



(c) Applied torques

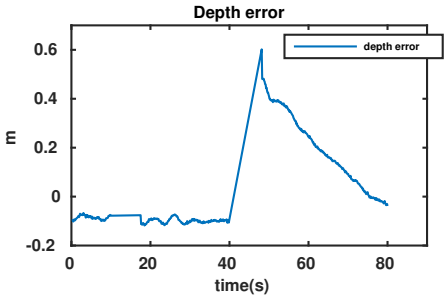


(d) PWM of thrusters

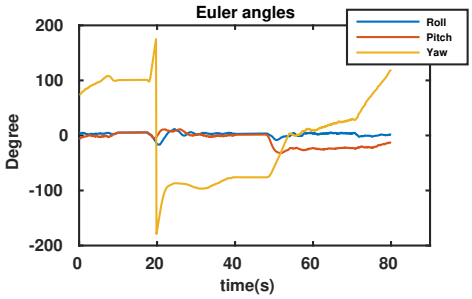
Figure 8.21 – Surge, pitch, and yaw control https://youtu.be/1DzfYrsSaMM and https://youtu.be/9eFT7h-zX3s

8.2.2 Integrated mission

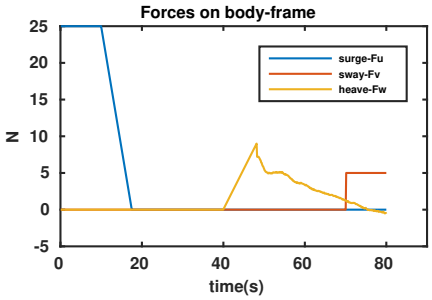
In this test, the robot carries out a complex task. Specifically, it goes straight (surge-pitch-yaw control) ($\alpha_F = \alpha_B = 45^0$), and then it changes configuration with $\alpha_F = 85^0$ and $\alpha_B = 45^0$ and make a turn 180^0. After that, it changes to a new configuration ($\alpha_F = 85^0$ and $\alpha_B = 85^0$) and does a depth control in a determinate time. Finally, robot holds the same configuration and controls in depth and sway directions simultaneously. Experiment results are shown in Figure 8.22.



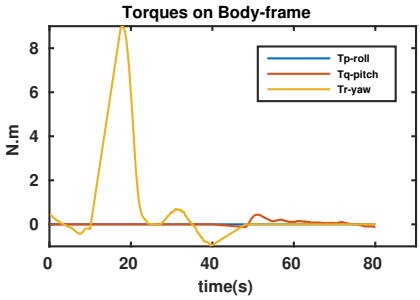
(a) Depth error



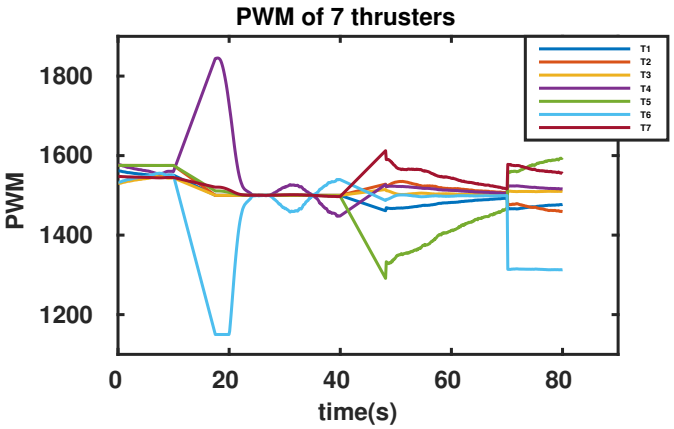
(b) Euler angles



(c) Forces in body-frame



(d) Torques in body-frame



(e) PWM of thrusters

Figure 8.22 – Integrated Mission of Umbrella Robot

8.3 Conclusion

This chapter presented simulation and experiment results of the Umbrella Robot. In simulation, we divide into two cases: reconfigurable and dynamic cases. The reconfigurable capability, shown by manipulability index and attainable force and torque spaces, are illustrated. This is considered as an extension of aforementioned discussions in Chapter 6. For dynamic configuration, different missions were studied from given desired control vector to dynamic desired control vector (output from a controller) which is designed for path following and station-keeping (observation) problem. Simulation results proved that our algorithm is efficient and feasible to implement in the real robot.

For experiments, the robot can carry out almost of basic missions such as Euler angles control, depth control, and surge control. As aforementioned discussions in chapter 6, the robot can modify its configuration with different missions and the most advantage is that the robot can go along surge direction like a torpedo robot which is very popular and has been used in a wide range of marine applications. A complex task with different configurations was tested and showed that the robot can accomplish a complex mission requiring an adaptive configuration system especially in karst exploration. The real test does not include dynamic configuration problem and it will be a future work.

Chapter 9

Conclusion, Perspective and future works

Contents

9.1 Conclusion and Perspective

As aforementioned, underwater robots are very important to discover underwater environment such as ocean, karst system, and its configuration can be classified as an under-actuated, fully-actuated, or over-actuated system depending on the relation between the number of actuators and DoFs. Redundant system is a system that generally speaking has more actuators than DoFs. For redundant systems, there are not only advantages but also disadvantages. Specifically, a redundant robot can be able to exploit this property for robust control, but it should yield higher cost and difficulties in control. For a redundant system, it is necessary to investigate control allocation method which is a mapping from desired force on each DoFs ($\mathbf{F}_B^d$) to desired force on each actuators ($\mathbf{F}_m^d$) and other relating perspectives such as control strategy, proper configuration which can leverage its redundancy.

The thesis exploited properties of redundant systems in propulsion perspective, particularly for static configuration design and reconfigurable one. For static configuration design, the problem is how to know position and orientation of thrusters with respect to some criteria. The thesis proposed performance criteria, i.e, manipulability index, energetic index, workspace index, reactive index and robustness index, for finding position and orientation of thrusters. A design process to optimize these criteria based on multiobjective optimization technique was proposed to solve the problem. Simulation results and experiments have been shown to prove our approach. Note that it is just one Pareto solution, and Pareto front investigation is promising and challenging study in the future. For a comparison, two configurations have been chosen for simulations and experiments on Cube robot, a redundant system with 8 thrusters installed at the corners of a cubic shape. The first one is a typical configuration in which thrusters are fixed vertically or horizontally, the second one is the optimal solution of our problem. The performance indices of Cube

robot, detailed in Chapter 5, with optimal configuration are better than ones with typical configuration. Optimal configuration of Cube robot is a good reference for designing underwater robots which especially carry out missions that require a fine turning of system reactivity (e.g. confined environments).

Motivated by adaptive robots, an autonomous underwater robot (AUV) with reconfigurable configuration has been designed from top to toe, called Umbrella Robot (UmRobot). The robot can be able to modify position and orientation of thrusters using two added DC motors. In this issue, the configuration matrix is not constant and can be changed depending on user's or mission's requirements. The details of designing, including hardware and software of AUV, were introduced in Chapter 6. Moreover, acting abilities along each DoFs, which measure how much a robot is able to act, were proposed to validate the robot's configuration. Note that they are different from performance criteria mentioned in the previous part. Basic and integrated experiments with the controller in Quaternion formalism were tested. They showed that UmRobot can be able to accomplish our missions in most of DoFs. Nevertheless, the robot also holds limitations in roll direction. An optimal configuration problem for UmRobot, with respect to geometric distances and initial angles of thrusters, was proposed and the new configuration showed that it is more isotropic than the previous one (current UmRobot). Furthermore, not only two robots have been built for simulations and experiments in the thesis, but also there are several underwater robots in EXPLORE team, a Matlab-based Toolbox was built to summarize performance indices and acting abilities of these robots.

Another idea of reconfigurable configuration was presented in Chapter 7, a proposed approach to optimize locally energy-like criterion of UmRobot with dynamic configuration. Our dynamic configuration problem depends on the output of controller, desired control vector $\mathbf{F}_B^d$, which is varying at each time step. This can be considered as multiparametric programming, however, this direction is out of scope of this thesis and will be a future work and discussed in the next section. Otherwise, the thesis proposed an algorithm, called A-SQP which is based on Sequential Quadratic Programming, to locally optimize energy-like criterion online. Because of online properties, one iteration is set up in proposed algorithm. Simulation results were shown to prove the efficiency of the proposed method. Several simulation cases were investigated. The first one is that the desired control vector $\mathbf{F}_B^d$ is given, the second one is that this vector is output from controller of path-following problem and observation problem (station-keeping). We can see that the UmRobot with dynamic configuration was more flexible to carry out missions and showed good performance w.r.t energy-like criterion. Moreover, for dynamic configuration, control allocation methods were also investigated and nonlinear programming based approaches show better performances than pseudo-inverse based methods.

An important perspective that we conclude is how to vary from optimal configuration of Cube robot ($\mathbf{C^2}$ configuration in Chapter 5) to Torpedo-like configuration of Cube robot and vice-versa. Indeed, for Cube robot, with the optimal configuration, performance indices and acting abilities along 6 DoFs are good. Torpedo-like configuration is very useful in some cases. An interesting idea for varying between

them can be seen in https://youtu.be/6vv4pxxIcyw. Note that the modification of robot's shape is not seriously considered in the video simulation. Beyond all, to increase robot's versatility, there exits other ways to improve its propulsion design and it will be an interesting future studies.

9.2 Future works

In fact, there remains potential directions for future works with redundant systems. For optimal static configuration design, other multiobjective optimization techniques can be investigated, especially heuristic methods. Besides, finding the set of Pareto solutions, Pareto front, is an interesting topic. However, it is not easy to do that for continuous problem. Other performance indices can be proposed in the design process and a global criterion, which is able to represent for a *"good"* robot, could be derived. A new Umbrella Robot with advanced configuration will be built to validate acting abilities in 6 DoFs and other control strategies in the future. Besides, a flexible mechanism for modifying configuration of Cube robot will be studied carefully.

For dynamic configuration problem, multiparametric programming can be applied for our optimal dynamic configuration to find the feasible set of desired control vector $\mathbf{F}_B^d$ which is a guidance for designing $\mathbf{F}_B^d$ in control layer. For online algorithm, contractivity and robustness will be studied. In case of switching between different configurations, a mechanism will be studied to make sure a smooth transition in control performances and other perspectives. Furthermore, an efficient control allocation method for bad points in configuration variation is also an interesting question. Experiments for dynamic configuration problem will be carry out to prove proposed approaches.

Appendix A

Appendix

A.1 Proofs and Mathematical basics

Theorem A.1.1 *The image of an unit hyper-sphere under any $n \times m$ matrix is a hyper-ellipsoid.*

Proof Let $\mathbf{A}$ be a $n \times m$ matrix with rank r. Let $\mathbf{A} = \mathbf{USV}^T$ be a singular value decomposition of $\mathbf{A}$. The left and right singular vectors of $\mathbf{A}$ are denoted as $\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n$ and $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_m$, respectively. Since $rank(A) = r$, the singular values of $\mathbf{A}$ have the properties: $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > 0$ and $\sigma_{r+1} = \sigma_{r+2} = ... = \sigma_m = 0$.

Let $\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix}$ be an unit vector in $\mathbb{R}^m$. Because $\mathbf{V}$ is an orthogonal matrix, and $\mathbf{V}^T$ is also, we have $\mathbf{V}^T\mathbf{x}$ is an unit vector (it is easy to see that $\|\mathbf{V}^T\mathbf{x}\| = \|\mathbf{x}\|$). So, $(\mathbf{v}_1^T\mathbf{x})^2 + (\mathbf{v}_2^T\mathbf{x})^2 + ... + (\mathbf{v}_m^T\mathbf{x})^2 = 1$.

On the other hand, we have $\mathbf{A} = \sigma_1\mathbf{u}_1\mathbf{v}_1^T + \sigma_2\mathbf{u}_2\mathbf{v}_2^T + ... + \sigma_r\mathbf{u}_r\mathbf{v}_r^T$. Therefore:

$$\begin{aligned} \mathbf{Ax} &= \sigma_1\mathbf{u}_1\mathbf{v}_1^T\mathbf{x} + \sigma_2\mathbf{u}_2\mathbf{v}_2^T\mathbf{x} + ... + \sigma_r\mathbf{u}_r\mathbf{v}_r^T\mathbf{x} \\ &= (\sigma_1\mathbf{v}_1^T\mathbf{x})\mathbf{u}_1 + (\sigma_2\mathbf{v}_2^T\mathbf{x})\mathbf{u}_2 + ... + (\sigma_r\mathbf{v}_r^T\mathbf{x})\mathbf{u}_r \\ &= \mathbf{y}_1\mathbf{u}_1 + \mathbf{y}_2\mathbf{u}_2 + ... + \mathbf{y}_r\mathbf{u}_r \\ &= \mathbf{Uy} \end{aligned} \tag{A.1}$$

where $\mathbf{y}_i$ denotes the $\sigma_i\mathbf{v}_i^T\mathbf{x}$, and $\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_r \end{pmatrix}$.

From (A.1), we have: $\|\mathbf{Ax}\| = \|\mathbf{Uy}\| = \|\mathbf{y}\|$ (since $\mathbf{U}$ is an orthogonal matrix). Moreover, $\mathbf{y}$ has the following property:

$$\begin{aligned} &(\frac{y_1}{\sigma_1})^2 + (\frac{y_2}{\sigma_2})^2 + ... + (\frac{y_r}{\sigma_r})^2 = \\ &= (\mathbf{v}_1^T\mathbf{x})^2 + (\mathbf{v}_2^T\mathbf{x})^2 + ... + (\mathbf{v}_r^T\mathbf{x})^2 \leq 1 \end{aligned} \tag{A.2}$$

Specifically:

1. If $r = m$ (of course, we must have $m \leq n$), the equality in equation (A.2) holds, and the image of unit hyper-sphere forms the surface of a hyper-ellipsoid.
2. If $r < m$, the image of unit hyper-sphere corresponds to a solid hyper-ellipsoid.

This completes the proof. ∎

Proposition A.1.2 *For given configuration matrix* $\mathbf{A}$*, directions of thrusters are the three first rows of the matrix and their positions can be found by Equation A.4.*

Proof Configuration matrix $\mathbf{A}$ has a form as:

$$\mathbf{A} = \begin{pmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_m \\ \boldsymbol{\tau}_1 & \boldsymbol{\tau}_2 & \cdots & \boldsymbol{\tau}_m \end{pmatrix} \tag{A.3}$$

It is obvious to see that direction of each thruster is determined by vector $\mathbf{u}_i$, and for each $\boldsymbol{\tau_i}$, position of each thruster can be interpolated as:

$$\mathbf{r}_i = \frac{\mathbf{u}_i \otimes \boldsymbol{\tau_i}}{\|\mathbf{u_i}\|^2} + k\mathbf{u_i}, k \in \mathbb{R} \tag{A.4}$$

with the assumption $\|\mathbf{r}_i\| = 1$, it is easy to find the coefficient k. ∎

A.2 Quaternions

A quaternion $\mathbf{Q}$ is a set of four parameters and can be written in different forms:

$$\mathbf{Q} = q_0 + q_1 i + q_2 j + q_3 k \tag{A.5}$$
$$\mathbf{Q} = (q_0, \mathbf{q}) \tag{A.6}$$
$$\mathbf{Q} = (q_0 \quad q_1 \quad q_2 \quad q_3)^T \tag{A.7}$$

A.2.1 Quaternion Operators

Consider two quaternions $\mathbf{P} = p_0 + p_1 i + p_2 j + p_3 k = (p_0, \mathbf{p})$ and $\mathbf{Q} = q_0 + q_1 i + q_2 j + q_3 k = (q_0, \mathbf{q})$, then, we have:

Addition:

$$\mathbf{P} + \mathbf{Q} = (p_0 + q_0) + (p_1 + q_1)i + (p_2 + q_2)j + (p_3 + q_3)k = (p_0 + q_0, \mathbf{p} + \mathbf{q}) \tag{A.8}$$

Multiplication:

$$\mathbf{P} \odot \mathbf{Q} = (p_0 q_0 - \mathbf{p}.\mathbf{q}, p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p} \times \mathbf{q}) \tag{A.9}$$

Conjugate:

$$\mathbf{Q}^* = q_0 - q_1 i - q_2 j - q_3 k = (q_0, -\mathbf{q}) \tag{A.10}$$

$$(\mathbf{Q} \odot \mathbf{Q})^* = \mathbf{Q}^* \odot \mathbf{P}^* \tag{A.11}$$

Norm:

$$\|\mathbf{Q}\| = \sqrt{\mathbf{Q}^*\mathbf{Q}} \tag{A.12}$$

A quaternion is called a unit quaternion if its norm is 1.

Inverse:

$$\mathbf{Q}^{-1} = \frac{\mathbf{Q}^*}{\|\mathbf{Q}\|^2} \tag{A.13}$$

it is easily to verify that $\mathbf{Q}\mathbf{Q}^{-1} = \mathbf{Q}^{-1}\mathbf{Q} = 1$. If $\mathbf{Q}$ is a unit quaternion, the inverse is its conjugate $\mathbf{Q}^*$.

Differentiation:

$$\frac{d}{dt}(\mathbf{P} \odot \mathbf{Q}) = \dot{\mathbf{P}} \odot \mathbf{Q} + \mathbf{P} \odot \dot{\mathbf{Q}} \tag{A.14}$$

A.2.2 Eulers angles to quaternions and vice versa

Rotation matrix from quaternions:

$$R = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_0q_2 + q_1q_3) \\ 2(q_1q_2 - q_0q_3) & (q_0^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_0q_1 + q_2q_3) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix} \tag{A.15}$$

Quaternions from Rotation matrix: The mapping from a rotation matrix to a quaternion is slightly complicated. This is found by solving a system of equations which may produce complex results. To avoid such an event, several composite functions, depending on the parameters of the rotation matrix, are defined to select the best of quaternion's four element. We can referred to [Diebel 2006] for more details.

Euler angles from quaternions:

$$\phi = \arctan \frac{-2(q_2q_3 - q_0q_1)}{q_0^2 - q_1^2 - q_2^2 + q_3^2} \tag{A.16}$$

$$\theta = \arcsin(2(q_0q_2 + q_1q_3)) \tag{A.17}$$

$$\psi = \arctan \frac{-2(q_1q_2 - q_0q_3)}{q_0^2 + q_1^2 - q_2^2 - q_3^2} \tag{A.18}$$

Quaternion from Euler angles: The order of rotation is ZYX

$$\mathbf{Q} = \begin{pmatrix} \cos(\psi/2)\cos(\theta/2)\cos(\phi/2) + \sin(\psi/2)\sin(\theta/2)\sin(\phi/2) \\ \cos(\psi/2)\cos(\theta/2)\sin(\phi/2) - \sin(\psi/2)\sin(\theta/2)\cos(\phi/2) \\ \cos(\psi/2)\sin(\theta/2)\cos(\phi/2) + \sin(\psi/2)\cos(\theta/2)\sin(\phi/2) \\ \sin(\psi/2)\cos(\theta/2)\cos(\phi/2) - \cos(\psi/2)\sin(\theta/2)\sin(\phi/2) \end{pmatrix} \tag{A.19}$$

A.3 Configuration matrix for Umbrella Robot

This section presents how to determine the configuration matrix, **A**, of Umbrella Robot. The notations used in the part is displayed in Table A.1 and Figures A.1 and A.2. Note that all elements **u**, **r** of configuration matrix are expressed in Body-frame. The method used is to rotate and to translate through points (**A**, **B**, **C**, **D**) consequently. Several temporal coordinate systems are denoted: $X_iY_iZ_i, i = 1, 2, 3$. A notation $\mathbf{D}_7^3$ is point **D** of thruster 7 model expressed in $X_3Y_3Z_3$ frame (B for Body-frame). A rotation matrix ${}^3\mathbf{R}_y^B$ is rotating action about Y axis (Body-frame) from $X_3Y_3Z_3$ frame to Body-frame (XYZ).

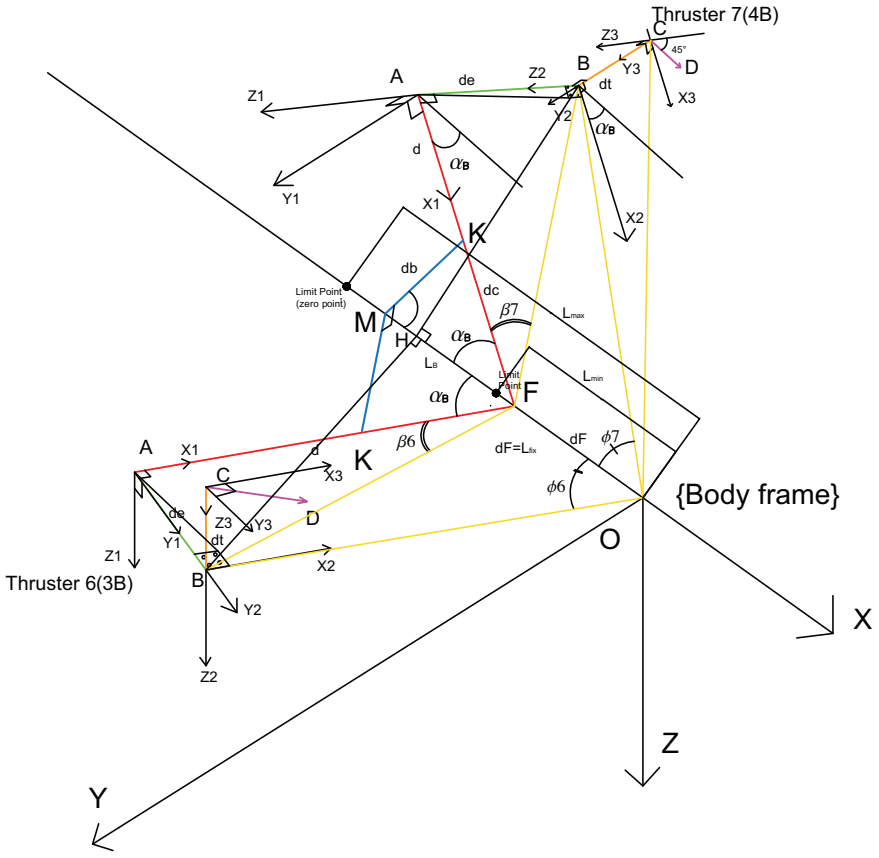


Figure A.1 – The geometry of thruster 7 and 6 for **A** matrix

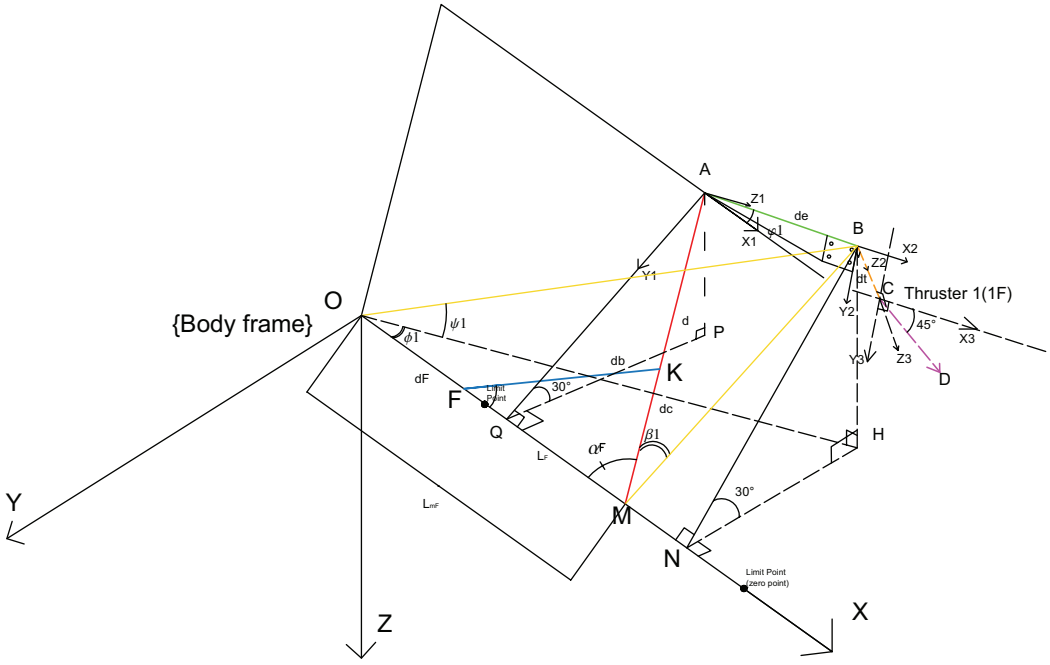


Figure A.2 – The geometry of thruster 1 for **A** matrix

$$\mathbf{u}_7^B = {}^3\mathbf{R}_y^B \mathbf{D}_7^3 = \begin{pmatrix} cos\alpha_B & 0 & -sin\alpha_B \\ 0 & 1 & 0 \\ sin\alpha_B & 0 & cos\alpha_B \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ -\frac{1}{\sqrt{2}} \end{pmatrix} \tag{A.20}$$

$$= \begin{pmatrix} \frac{cos\alpha_B + sin\alpha_B}{\sqrt{2}} \\ 0 \\ \frac{-cos\alpha_B + sin\alpha_B}{\sqrt{2}} \end{pmatrix} \tag{A.21}$$

This step computes the vector $\mathbf{r}_7^B$:

$$\begin{aligned} \mathbf{r}_7^B = \mathbf{C}_7^B &= {}^2\mathbf{R}_y^B \mathbf{C}_7^2 + \mathbf{B}_7^B \\ &= \begin{pmatrix} cos\alpha_B & 0 & -sin\alpha_B \\ 0 & 1 & 0 \\ sin\alpha_B & 0 & cos\alpha_B \end{pmatrix} \begin{pmatrix} 0 \\ -d_t \\ 0 \end{pmatrix} + \begin{pmatrix} x_{\mathbf{B}7}^B \\ y_{\mathbf{B}7}^B \\ z_{\mathbf{B}7}^B \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ -d_t \\ 0 \end{pmatrix} + \begin{pmatrix} x_{\mathbf{B}7}^B \\ 0 \\ z_{\mathbf{B}7}^B \end{pmatrix} = \begin{pmatrix} x_{\mathbf{B}7}^B \\ -d_t \\ z_{\mathbf{B}7}^B \end{pmatrix} \end{aligned} \tag{A.22}$$

where $(x_{\mathbf{B}7}^B \quad y_{\mathbf{B}7}^B \quad z_{\mathbf{B}7}^B)^T$ is coordinate vector of point **B** of thruster 7 in Body-frame.

No.	Notation	Description	Units
1	α_F	the forward angle	degree
2	α_B	the backward angle	degree
3	d	the length of a rib of the umbrella	mm
4	d_e	the geometric distance of part holding thruster	mm
5	d_t	the geometric distance of part holding thruster	mm
6	d_F	the distance from origin of body frame to notch(fixed point)	mm
7	L_F	the forward distance from notch(fixed point) to runner(moving point)	mm
8	L_B	the backward distance from notch(fixed point) to runner(moving point)	mm
9	$\beta_i (i = \overline{1, ..., 7})$	the auxiliary angle	degree

Table A.1 – Notations in the umbrella robot scheme

We have:

$$\begin{aligned} x^B_{\mathbf{B}7} &= -OBcos\phi_7 = -OH \\ &= -(OF \pm FH) = -(d_F \pm BF.cos(\alpha_B + \beta_7)) \\ &= -(d_F \pm \sqrt{d_e^2 + d^2}.cos(\alpha_B + \beta_7)) \\ z^B_{\mathbf{B}7} &= -OBsin\phi_7 = -BH = -BF.sin(\alpha_B + \beta_7) \\ &= -\sqrt{d_e^2 + d^2}.sin(\alpha_B + \beta_7) \end{aligned} \tag{A.23}$$

From (A.22) and (A.23), we have:

$$\mathbf{r}^B_7 = \begin{pmatrix} -(d_F \pm \sqrt{d^2 + d_e^2} cos(\alpha_B + \beta_7)) \\ -d_t \\ -(\sqrt{d^2 + d_e^2} sin(\alpha_B + \beta_7)) \end{pmatrix} \tag{A.24}$$

In case of $\beta_7 \approx 0$ and $d_e \approx 0$, the (A.24) becomes:

$$\mathbf{r}^B_7 = \begin{pmatrix} -(d_F \pm dcos\alpha_B) \\ -d_t \\ -dsin\alpha_B \end{pmatrix} \tag{A.25}$$

In similarity, the vector $\mathbf{u}_6^B$ and $\mathbf{r}_6^B$ are computed as follows:

$$\begin{aligned}\mathbf{u}_6^B &= {}^3\mathbf{R}_z^B\mathbf{D}_6^3 = \begin{pmatrix} cos\alpha_B & sin\alpha_B & 0 \\ -sin\alpha_B & cos\alpha_B & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} \frac{cos\alpha_B + sin\alpha_B}{\sqrt{2}} \\ \frac{cos\alpha_B - sin\alpha_B}{\sqrt{2}} \\ 0 \end{pmatrix}\end{aligned} \tag{A.26}$$

The vector $\mathbf{r}_6^B$ is computed as:

$$\begin{aligned}\mathbf{r}_6^B &= \mathbf{C}_6^B = {}^2\mathbf{R}_z^B\mathbf{C}_6^2 + \mathbf{B}_6^B \\ &= \begin{pmatrix} cos\alpha_B & sin\alpha_B & 0 \\ -sin\alpha_B & cos\alpha_B & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ -d_t \end{pmatrix} + \begin{pmatrix} x_{\mathbf{B}6}^B \\ y_{\mathbf{B}6}^B \\ z_{\mathbf{B}6}^B \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ 0 \\ -d_t \end{pmatrix} + \begin{pmatrix} x_{\mathbf{B}6}^B \\ y_{\mathbf{B}6}^B \\ 0 \end{pmatrix} = \begin{pmatrix} x_{\mathbf{B}6}^B \\ y_{\mathbf{B}6}^B \\ -d_t \end{pmatrix}\end{aligned} \tag{A.27}$$

We have:

$$\begin{aligned}x_{\mathbf{B}6}^B &= -OBcos\phi_6 = -OH \\ &= -(OF \pm FH) = -(d_F \pm BF.cos(\alpha_B + \beta_6)) \\ &= -(d_F \pm \sqrt{d_e^2 + d^2}.cos(\alpha_B + \beta_6)) \\ y_{\mathbf{B}6}^B &= OBsin\phi_6 = BH = BF.sin(\alpha_B + \beta_6) \\ &= \sqrt{d_e^2 + d^2}.sin(\alpha_B + \beta_6)\end{aligned} \tag{A.28}$$

From (A.27) and (A.28), we have:

$$\mathbf{r}_6^B = \begin{pmatrix} -(d_F \pm \sqrt{d^2 + d_e^2}cos(\alpha_B + \beta_6)) \\ \sqrt{d^2 + d_e^2}sin(\alpha_B + \beta_6) \\ -d_t \end{pmatrix} \tag{A.29}$$

In case of $\beta_6 \approx 0$ and $d_e \approx 0$, the (A.29) becomes:

$$\mathbf{r}_6^B = \begin{pmatrix} -(d_F \pm dcos\alpha_B) \\ dsin\alpha_B \\ -d_t \end{pmatrix} \tag{A.30}$$

The vector $\mathbf{u}_4^B$ and $\mathbf{r}_4^B$ are computed as follows:

$$\mathbf{u}_4^B = {}^3\mathbf{R}_z^B\mathbf{D}_4^3 = \begin{pmatrix} cos\alpha_B & -sin\alpha_B & 0 \\ sin\alpha_B & cos\alpha_B & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}$$
$$= \begin{pmatrix} \frac{cos\alpha_B + sin\alpha_B}{\sqrt{2}} \\ \frac{-cos\alpha_B + sin\alpha_B}{\sqrt{2}} \\ 0 \end{pmatrix} \tag{A.31}$$

The vector $\mathbf{r}_4^B$ is computed as:

$$\mathbf{r}_4^B = \mathbf{C}_4^B = {}^2\mathbf{R}_z^B\mathbf{C}_4^2 + \mathbf{B}_4^B$$
$$= \begin{pmatrix} cos\alpha_B & -sin\alpha_B & 0 \\ sin\alpha_B & cos\alpha_B & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ -d_t \end{pmatrix} + \begin{pmatrix} x_{\mathbf{B}4}^B \\ y_{\mathbf{B}4}^B \\ z_{\mathbf{B}4}^B \end{pmatrix}$$
$$= \begin{pmatrix} 0 \\ 0 \\ -d_t \end{pmatrix} + \begin{pmatrix} x_{\mathbf{B}4}^B \\ y_{\mathbf{B}4}^B \\ 0 \end{pmatrix} = \begin{pmatrix} x_{\mathbf{B}4}^B \\ y_{\mathbf{B}4}^B \\ -d_t \end{pmatrix} \tag{A.32}$$

We have:

$$x_{\mathbf{B}4}^B = -OB.cos\phi_4 = -OH$$
$$= -(OF \pm FH) = -(d_F \pm BF.cos(\alpha_B + \beta_4))$$
$$= -(d_F \pm \sqrt{d_e^2 + d^2}.cos(\alpha_B + \beta_4))$$
$$y_{\mathbf{B}4}^B = -OB.sin\phi_4 = -BH = -BF.sin(\alpha_B + \beta_4)$$
$$= -\sqrt{d_e^2 + d^2}sin(\alpha_B + \beta_4) \tag{A.33}$$

Therefore:

$$\mathbf{r}_4^B = \begin{pmatrix} -(d_F \pm \sqrt{d^2 + d_e^2}cos(\alpha_B + \beta_4)) \\ -(\sqrt{d^2 + d_e^2}sin(\alpha_B + \beta_4)) \\ -dt \end{pmatrix} \tag{A.34}$$

In case of $\beta_4 \approx 0$ and $d_e \approx 0$, the equation (A.34) becomes:

$$\mathbf{r}_4^B = \begin{pmatrix} -(d_F \pm dcos\alpha_B) \\ -dsin\alpha_B \\ -dt \end{pmatrix} \tag{A.35}$$

The vectors $\mathbf{u}_5^B$, $\mathbf{r}_5^B$ are computed as follows:

$$\mathbf{u}_5^B = {}^3\mathbf{R}_y^B\mathbf{D}_5^3 = \begin{pmatrix} cos\alpha_B & 0 & sin\alpha_B \\ 0 & 1 & 0 \\ -sin\alpha_B & 0 & cos\alpha_B \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix} \tag{A.36}$$
$$= \begin{pmatrix} \frac{cos\alpha_B + sin\alpha_B}{\sqrt{2}} \\ 0 \\ \frac{cos\alpha_B - sin\alpha_B}{\sqrt{2}} \end{pmatrix} \tag{A.37}$$

We have:

$$\begin{aligned}\mathbf{r}_5^B &= \mathbf{C}_5^B = {}^2\mathbf{R}_y^B\mathbf{C}_5^2 + \mathbf{B}_5^B \\ &= \begin{pmatrix} cos\alpha_B & 0 & sin\alpha_B \\ 0 & 1 & 0 \\ -sin\alpha_B & 0 & cos\alpha_B \end{pmatrix}\begin{pmatrix} 0 \\ d_t \\ 0 \end{pmatrix} + \begin{pmatrix} x_{\mathbf{B}5}^B \\ y_{\mathbf{B}5}^B \\ z_{\mathbf{B}5}^B \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ d_t \\ 0 \end{pmatrix} + \begin{pmatrix} x_{\mathbf{B}5}^B \\ 0 \\ z_{\mathbf{B}5}^B \end{pmatrix} = \begin{pmatrix} x_{\mathbf{B}5}^B \\ d_t \\ z_{\mathbf{B}5}^B \end{pmatrix}\end{aligned} \tag{A.38}$$

In $OXYZ$ coordinate system:

$$\begin{aligned}x_{\mathbf{B}5}^B &= -OB.cos\phi_5 = -OH = -(OF \pm FH) \\ &= -(d_F \pm BF.cos(\alpha_B + \beta_5)) \\ &= -(d_F \pm \sqrt{d_e^2 + d^2}.cos(\alpha_B + \beta_5)) \\ z_{\mathbf{B}5}^B &= OB.sin\phi_5 = BH = BF.sin(\alpha_B + \beta_5) \\ &= \sqrt{d_e^2 + d^2}.sin(\alpha_B + \beta_5)\end{aligned} \tag{A.39}$$

Therefore:

$$\mathbf{r}_5^B = \begin{pmatrix} -(d_F \pm \sqrt{d^2 + d_e^2}cos(\alpha_B + \beta_5)) \\ d_t \\ \sqrt{d^2 + d_e^2}sin(\alpha_B + \beta_5) \end{pmatrix} \tag{A.40}$$

In case of $\beta_5 \approx 0$ and $d_e \approx 0$, the equation (A.40) becomes:

$$\mathbf{r}_5^B = \begin{pmatrix} -(d_F \pm dcos\alpha_B) \\ d_t \\ dsin\alpha_B \end{pmatrix} \tag{A.41}$$

The following section presents the calculations of forward side of the umbrella robot.

The vector $\mathbf{u}_2^B$ and $\mathbf{r}_2^B$ are computed as:

$$\begin{aligned}\mathbf{u}_2^B = {}^3\mathbf{R}_y^B\mathbf{D}_2^3 &= \begin{pmatrix} cos\alpha_F & 0 & sin\alpha_F \\ 0 & 1 & 0 \\ -sin\alpha_F & 0 & cos\alpha_F \end{pmatrix}\begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix} \\ &= \begin{pmatrix} \frac{cos\alpha_F + sin\alpha_F}{\sqrt{2}} \\ 0 \\ \frac{cos\alpha_F - sin\alpha_F}{\sqrt{2}} \end{pmatrix}\end{aligned} \tag{A.42}$$

The vector $\mathbf{r}_2^B$ is computed as follows:

$$\begin{aligned}\mathbf{r}_2^B = \mathbf{C}_2^B &= {}^2\mathbf{R}_y^B\mathbf{C}_2^2 + \mathbf{B}_2^B \\ &= \begin{pmatrix} cos\alpha_F & 0 & sin\alpha_F \\ 0 & 1 & 0 \\ -sin\alpha_F & 0 & cos\alpha_F \end{pmatrix}\begin{pmatrix} 0 \\ d_t \\ 0 \end{pmatrix} + \begin{pmatrix} x_{\mathbf{B}2}^B \\ y_{\mathbf{B}2}^B \\ z_{\mathbf{B}2}^B \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ d_t \\ 0 \end{pmatrix} + \begin{pmatrix} x_{\mathbf{B}2}^B \\ 0 \\ z_{\mathbf{B}2}^B \end{pmatrix} = \begin{pmatrix} x_{\mathbf{B}2}^B \\ d_t \\ z_{\mathbf{B}2}^B \end{pmatrix}\end{aligned} \tag{A.43}$$

We have:

$$\begin{aligned} x_{\mathbf{B}2}^B &= OB.cos\phi_2 = ON = OM \mp NM \\ &= (d_F + L_F) \mp BM.cos(\alpha_F + \beta_2) \\ &= (d_F + L_F) \mp \sqrt{d_e^2 + d^2}.cos(\alpha_F + \beta_2) \\ z_{\mathbf{B}2}^B &= OB.sin\phi_2 = BN = BM.sin(\alpha_F + \beta_2) \\ &= \sqrt{d_e^2 + d^2}.sin(\alpha_F + \beta_2) \end{aligned} \tag{A.44}$$

Therefore:

$$\mathbf{r}_2^B = \begin{pmatrix} (d_F + L_F) \mp \sqrt{d^2 + d_e^2}cos(\alpha_F + \beta_2)) \\ d_t \\ \sqrt{d^2 + d_e^2}sin(\alpha_F + \beta_2) \end{pmatrix} \tag{A.45}$$

In case of $\beta_2 \approx 0$ and $d_e \approx 0$, the equation (A.45) becomes:

$$\mathbf{r}_2^B = \begin{pmatrix} d_F + L_F \mp dcos\alpha_F \\ d_t \\ dsin\alpha_F \end{pmatrix} \tag{A.46}$$

The vector $\mathbf{u}_1^B$ and $\mathbf{r}_1^B$ are computed as follows:

$$\begin{aligned}\mathbf{u}_1^B = {}^3\mathbf{R}_{zx}^B\mathbf{D}_1^3 &= {}^3\mathbf{R}_x^B(30^0){}^3\mathbf{R}_z^B(\varphi_1)\mathbf{D}_1^3 \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(30^0) & -sin(30^0) \\ 0 & sin(30^0) & cos(30^0) \end{pmatrix}\begin{pmatrix} cos\varphi_1 & sin\varphi_1 & 0 \\ -sin\varphi_1 & cos\varphi_1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} cos\varphi_1 & sin\varphi_1 & 0 \\ -\frac{\sqrt{3}}{2}sin\varphi_1 & \frac{\sqrt{3}}{2}cos\varphi_1 & -\frac{1}{2} \\ -\frac{1}{2}sin\varphi_1 & \frac{1}{2}cos\varphi_1 & \frac{\sqrt{3}}{2} \end{pmatrix}\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{\sqrt{2}}(cos\varphi_1 + sin\varphi_1) \\ \frac{\sqrt{3}}{2\sqrt{2}}(-sin\varphi_1 + cos\varphi_1) \\ \frac{1}{2\sqrt{2}}(-sin\varphi_1 + cos\varphi_1) \end{pmatrix}\end{aligned} \tag{A.47}$$

where $\varphi_1 = 90^0 - \alpha_F$

Therefore, the vector $\mathbf{u}_1^B$ becomes:

$$\mathbf{u}_1^B = \begin{pmatrix} \frac{1}{\sqrt{2}}(sin\alpha_F + cos\alpha_F) \\ \frac{\sqrt{3}}{2\sqrt{2}}(-cos\alpha_F + sin\alpha_F) \\ \frac{1}{2\sqrt{2}}(-cos\alpha_F + sin\alpha_F) \end{pmatrix} \tag{A.48}$$

The vector $\mathbf{r}_1^B$ is calculated as follows:

$$\begin{aligned} \mathbf{r}_1^B &= {}^2\mathbf{R}_{zx}^B \mathbf{C}_1^2 + \mathbf{B}_1^B \\ &= \begin{pmatrix} cos\varphi_1 & sin\varphi_1 & 0 \\ -\frac{\sqrt{3}}{2}sin\varphi_1 & \frac{\sqrt{3}}{2}cos\varphi_1 & -\frac{1}{2} \\ -\frac{1}{2}sin\varphi_1 & \frac{1}{2}cos\varphi_1 & \frac{\sqrt{3}}{2} \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ d_t \end{pmatrix} + \begin{pmatrix} x_{\mathbf{B}1}^B \\ y_{\mathbf{B}1}^B \\ z_{\mathbf{B}1}^B \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ -\frac{d_t}{2} \\ \frac{\sqrt{3}}{2}d_t \end{pmatrix} + \begin{pmatrix} x_{\mathbf{B}1}^B \\ y_{\mathbf{B}1}^B \\ z_{\mathbf{B}1}^B \end{pmatrix} = \begin{pmatrix} x_{\mathbf{B}1}^B(\alpha_F) \\ -\frac{d_t}{2} + y_{\mathbf{B}1}^B(\alpha_F) \\ \frac{\sqrt{3}}{2}d_t + z_{\mathbf{B}1}^B(\alpha_F) \end{pmatrix} \end{aligned} \tag{A.49}$$

We have:

$$\begin{aligned} x_{\mathbf{B}1}^B &= OB.cos\psi.cos\phi_1 = ON = OM \mp NM \\ &= (d_F + L_F) \mp BM.cos(\alpha_F + \beta_1) \\ &= (d_F + L_F) \mp \sqrt{d_e^2 + d^2}.cos(\alpha_F + \beta_1) \\ y_{\mathbf{B}1}^B &= -OB.cos\psi.sin\phi_1 = -NH = -BN.cos30^0 \\ &= -BM.sin(\alpha_F + \beta_1)\frac{\sqrt{3}}{2} = -\frac{\sqrt{3(d_e^2 + d^2)}sin(\alpha_F + \beta_1)}{2} \\ z_{\mathbf{B}1}^B &= -OB.sin\psi = -BH = -BN.sin30^0 \\ &= -BM.sin(\alpha_F + \beta_1).\frac{1}{2} = -\frac{\sqrt{d_e^2 + d^2}sin(\alpha_F + \beta_1)}{2} \end{aligned} \tag{A.50}$$

So, the vector $\mathbf{r}_1^B$ becomes:

$$\mathbf{r_1^B} = \begin{pmatrix} (d_F + L_F) \mp \sqrt{d_e^2 + d^2}.cos(\alpha_F + \beta_1) \\ -\frac{d_t}{2} - \frac{\sqrt{3(d_e^2+d^2)}sin(\alpha_F+\beta_1)}{2} \\ \frac{\sqrt{3}}{2}d_t - \frac{\sqrt{d_e^2+d^2}sin(\alpha_F+\beta_1)}{2} \end{pmatrix} \tag{A.51}$$

In case of $\beta_1 \approx 0$ and $d_e \approx 0$, the equation (A.51) becomes:

$$\mathbf{r_1^B} = \begin{pmatrix} (d_F + L_F) \mp d.cos\alpha_F \\ -\frac{d_t}{2} - \frac{\sqrt{3}d.sin\alpha_F}{2} \\ \frac{\sqrt{3}}{2}d_t - \frac{d.sin\alpha_F}{2} \end{pmatrix} \tag{A.52}$$

In similarity, the vector $\mathbf{u}_3^B$ and $\mathbf{r}_3^B$ are computed as follows:

$$\mathbf{u}_3^B = {}^3\mathbf{R}_{zx}^B\mathbf{D}_3^3 = {}^3\mathbf{R}_x^B(30^0){}^3\mathbf{R}_z^B(\varphi_3)\mathbf{D}_3^3 \tag{A.53}$$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(30^0) & sin(30^0) \\ 0 & -sin(30^0) & cos(30^0) \end{pmatrix} \begin{pmatrix} cos\varphi_3 & -sin\varphi_3 & 0 \\ sin\varphi_3 & cos\varphi_3 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} cos\varphi_3 & -sin\varphi_3 & 0 \\ \frac{\sqrt{3}}{2}sin\varphi_3 & \frac{\sqrt{3}}{2}cos\varphi_3 & \frac{1}{2} \\ -\frac{1}{2}sin\varphi_3 & -\frac{1}{2}cos\varphi_3 & \frac{\sqrt{3}}{2} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{\sqrt{2}}(cos\varphi_3 + sin\varphi_3) \\ \frac{\sqrt{3}}{2\sqrt{2}}(sin\varphi_3 - cos\varphi_3) \\ \frac{1}{2\sqrt{2}}(cos\varphi_3 - sin\varphi_3) \end{pmatrix} \tag{A.54}$$

where $\varphi_3 = 90 - \alpha_F$

Therefore, the vector $\mathbf{u}_3^B$ becomes:

$$\mathbf{u}_3^B = \begin{pmatrix} \frac{1}{\sqrt{2}}(sin\alpha_F + cos\alpha_F) \\ \frac{\sqrt{3}}{2\sqrt{2}}(cos\alpha_F - sin\alpha_F) \\ \frac{1}{2\sqrt{2}}(sin\alpha_F - cos\alpha_F) \end{pmatrix} \tag{A.55}$$

The vector $\mathbf{r}_3^B$ is computed as:

$$\mathbf{r}_3^B = {}^2\mathbf{R}_{zx}^B\mathbf{C}_3^2 + \mathbf{B}_3^B$$

$$= \begin{pmatrix} cos\varphi_3 & -sin\varphi_3 & 0 \\ \frac{\sqrt{3}}{2}sin\varphi_3 & \frac{\sqrt{3}}{2}cos\varphi_3 & \frac{1}{2} \\ -\frac{1}{2}sin\varphi_3 & -\frac{1}{2}cos\varphi_3 & \frac{\sqrt{3}}{2} \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ d_t \end{pmatrix} + \begin{pmatrix} x_{\mathbf{B}3}^B \\ y_{\mathbf{B}3}^B \\ z_{\mathbf{B}3}^B \end{pmatrix}$$

$$= \begin{pmatrix} 0 \\ \frac{d_t}{2} \\ \frac{\sqrt{3}}{2}d_t \end{pmatrix} + \begin{pmatrix} x_{\mathbf{B}3}^B \\ y_{\mathbf{B}3}^B \\ z_{\mathbf{B}3}^B \end{pmatrix} = \begin{pmatrix} x_{\mathbf{B}3}^B \\ \frac{d_t}{2} + y_{\mathbf{B}3}^B \\ \frac{\sqrt{3}}{2}d_t + z_{\mathbf{B}3}^B \end{pmatrix} \tag{A.56}$$

We have:

$$x_{\mathbf{B}3}^B = ON = OM \mp NM = (d_F + L_F) \mp BM.cos(\alpha_F + \beta_3)$$

$$= (d_F + L_F) \mp \sqrt{d_e^2 + d^2}.cos(\alpha_F + \beta_3) \tag{A.57}$$

$$y_{\mathbf{B}3}^B = NH = BN.cos(30^0)$$

$$= BM.sin(\alpha_F + \beta_3).cos(30^0)$$

$$= \frac{\sqrt{3(d_e^2 + d^2)}}{2}.sin(\alpha_F + \beta_3) \tag{A.58}$$

$$z_{\mathbf{B}3}^B = -BH = -BN.sin(30^0)$$

$$= -BM.sin(\alpha_F + \beta_3).sin(30^0)$$

$$= -\frac{\sqrt{d_e^2 + d^2}}{2}sin(\alpha_F + \beta_3) \tag{A.59}$$

So, the vector $\mathbf{r}_3^B$ is:

$$\mathbf{r}_3^B = \begin{pmatrix} (d_F + L_F) \mp \sqrt{d_e^2 + d^2}.cos(\alpha_F + \beta_3) \\ \frac{d_t}{2} + \frac{\sqrt{3(d_e^2+d^2)}}{2}.sin(\alpha_F + \beta_3) \\ \frac{\sqrt{3}}{2}d_t - \frac{\sqrt{d_e^2+d^2}}{2}sin(\alpha_F + \beta_3) \end{pmatrix} \tag{A.60}$$

In case of $\beta_3 \approx 0$ and $d_e \approx 0$, the equation (A.60) becomes:

$$\mathbf{r}_3^B = \begin{pmatrix} (d_F + L_F) \mp d.cos\alpha_F \\ \frac{d_t}{2} + \frac{\sqrt{3}}{2}.d.sin\alpha_F \\ \frac{\sqrt{3}}{2}d_t - \frac{d}{2}sin\alpha_F \end{pmatrix} \tag{A.61}$$

In summary:

$\mathbf{u}_1^B = \begin{pmatrix} \frac{1}{\sqrt{2}}(sin\alpha_F + cos\alpha_F) \\ \frac{\sqrt{3}}{2\sqrt{2}}(-cos\alpha_F + sin\alpha_F) \\ \frac{1}{2\sqrt{2}}(-cos\alpha_F + sin\alpha_F) \end{pmatrix}$	$\mathbf{r_1^B} = \begin{pmatrix} (d_F + L_F) - \sqrt{d_e^2 + d^2}.cos(\alpha_F + \beta_1) \\ -\frac{d_t}{2} - \frac{\sqrt{3(d_e^2+d^2)}sin(\alpha_F+\beta_1)}{2} \\ \frac{\sqrt{3}}{2}d_t - \frac{\sqrt{d_e^2+d^2}sin(\alpha_F+\beta_1)}{2} \end{pmatrix}$
$\mathbf{u}_2^B = \begin{pmatrix} \frac{cos\alpha_F + sin\alpha_F}{\sqrt{2}} \\ 0 \\ \frac{cos\alpha_F - sin\alpha_F}{\sqrt{2}} \end{pmatrix}$	$\mathbf{r}_2^B = \begin{pmatrix} (d_F + L_F) - \sqrt{d^2 + d_e^2}cos(\alpha_F + \beta_2)) \\ d_t \\ \sqrt{d^2 + d_e^2}sin(\alpha_F + \beta_2) \end{pmatrix}$
$\mathbf{u}_3^B = \begin{pmatrix} \frac{1}{\sqrt{2}}(sin\alpha_F + cos\alpha_F) \\ \frac{\sqrt{3}}{2\sqrt{2}}(cos\alpha_F - sin\alpha_F) \\ \frac{1}{2\sqrt{2}}(sin\alpha_F - cos\alpha_F) \end{pmatrix}$	$\mathbf{r}_3^B = \begin{pmatrix} (d_F + L_F) - \sqrt{d_e^2 + d^2}.cos(\alpha_F + \beta_3) \\ \frac{d_t}{2} + \frac{\sqrt{3(d_e^2+d^2)}}{2}.sin(\alpha_F + \beta_3) \\ \frac{\sqrt{3}}{2}d_t - \frac{\sqrt{d_e^2+d^2}}{2}sin(\alpha_F + \beta_3) \end{pmatrix}$
$\mathbf{u}_4^B = \begin{pmatrix} \frac{cos\alpha_B + sin\alpha_B}{\sqrt{2}} \\ \frac{-cos\alpha_B + sin\alpha_B}{\sqrt{2}} \\ 0 \end{pmatrix}$	$\mathbf{r}_4^B = \begin{pmatrix} -(d_F + \sqrt{d^2 + d_e^2}cos(\alpha_B + \beta_4)) \\ -(\sqrt{d^2 + d_e^2}sin(\alpha_B + \beta_4)) \\ -dt \end{pmatrix}$
$\mathbf{u}_5^B = \begin{pmatrix} \frac{cos\alpha_B + sin\alpha_B}{\sqrt{2}} \\ 0 \\ \frac{cos\alpha_B - sin\alpha_B}{\sqrt{2}} \end{pmatrix}$	$\mathbf{r}_5^B = \begin{pmatrix} -(d_F + \sqrt{d^2 + d_e^2}cos(\alpha_B + \beta_5)) \\ d_t \\ \sqrt{d^2 + d_e^2}sin(\alpha_B + \beta_5) \end{pmatrix}$
$\mathbf{u}_6^B = \begin{pmatrix} \frac{cos\alpha_B + sin\alpha_B}{\sqrt{2}} \\ \frac{cos\alpha_B - sin\alpha_B}{\sqrt{2}} \\ 0 \end{pmatrix}$	$\mathbf{r}_6^B = \begin{pmatrix} -(d_F + \sqrt{d^2 + d_e^2}cos(\alpha_B + \beta_6)) \\ \sqrt{d^2 + d_e^2}sin(\alpha_B + \beta_6) \\ -d_t \end{pmatrix}$
$\mathbf{u}_7^B = \begin{pmatrix} \frac{cos\alpha_B + sin\alpha_B}{\sqrt{2}} \\ 0 \\ \frac{-cos\alpha_B + sin\alpha_B}{\sqrt{2}} \end{pmatrix}$	$\mathbf{r}_7^B = \begin{pmatrix} -(d_F + \sqrt{d^2 + d_e^2}cos(\alpha_B + \beta_7)) \\ -d_t \\ -(\sqrt{d^2 + d_e^2}sin(\alpha_B + \beta_7)) \end{pmatrix}$

Table A.2 – Elements of **A** matrix

A.4 Modeling of Umbrella Robot by numerical simulations

For dynamic parameters of the underwater robot, we can have the mass by weighting the robot. For moments of inertia, SOLIDWORKS is used to estimate. In our simulation, a simple model was considered: **M** (rigid-body mass matrix), **D** (damping matrix) are diagonal matrices. **C** (Corriolis-Centripetal matrix) and

g (hydrostatic matrix) are neglected. Note that added mass and external forces (current, wind and so on) are also not considered in the simulations.

The damping coefficients are estimated by ANSYS-CFX software which is used for Computational Fluid Dynamics (CFD). As most of CFD softwares, ANSYS-CFX solves a system of partial differential equations by discreting or transforming into a set of algebraic equations which can be solved digitally. Three most popular methods for this discretization are: Finite Element Method (FEM), Finite Volume Method (FVM), and Finite Difference Method (FDM). In general, FDM method is suitable for simple geometries, and FEM and FDM are chosen for more complex geometries. ANSYS-CFX uses FVM for computing fluid dynamics. The principle of CFD simulation can be summarized as:

1. A CAD model is first divided into very small but finite-sized elements of geometrically simple shapes (Meshing).
2. Partial differential equations and boundary conditions are set up for each small element.
3. Iterative methods are used to solve these equations (Convergence conditions).
4. Post-process for results.

Therefore, the procedure of estimating hydrodynamics parameters of the umbrella robot is divided into five steps:

1. Create a simplified 3D model the robot as well as the water tank for hydrodynamic analysis.
2. Mesh the robot model and the water tank for the FVM (Finite Volume Method) computation.
3. Set the boundary and stop conditions for the FVM computation.
4. Execute iterative FVM calculation until the convergence.
5. Post-process the FVM data to obtain the damping coefficients.

Figure A.3 shows the umbrella robot in ANSYS-CFX simulation with simplified 3D model, water tank, and velocity of fluid.

A.5 Path following methods

Recall that $\{NED\}$-frame is also denoted as $\{I\}$-frame.

A.5.1 Line of Sight

This part shows the principles of line of sight method for path following which could be straight-line path or curved one. We investigate the first case for straight-line path.

Straight-line path following: A straight-line is modeled by way-points. Suppose that we have a straight-line defined by two way-points $\mathbf{P}_{p_k}(x_{p_k}, y_{p_k})$ and

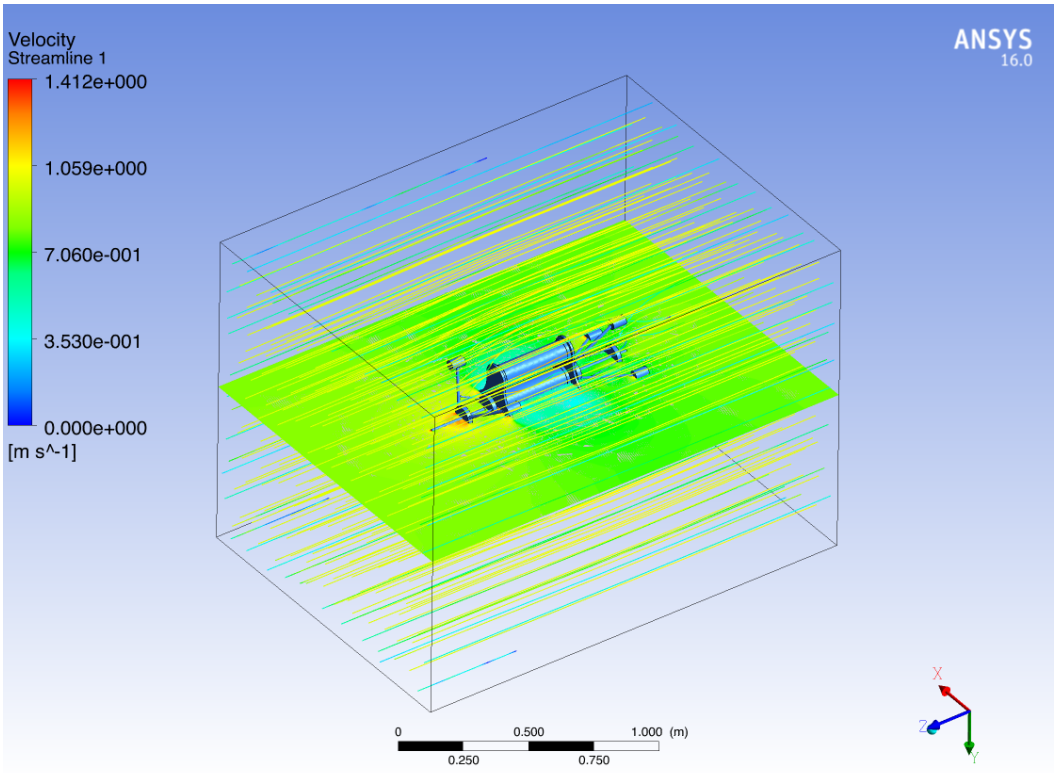


Figure A.3 – Umbrella robot simulation in ANSYS

$\mathbf{P}_{p_{k+1}}(x_{p_{k+1}}, y_{p_{k+1}})$ (expressed in $\{I\}$-frame) as in Figure A.4. The objective of guidance system is to find ψ^d (desired yaw angle/desired heading angle) which allows the vehicle following a path (assume that the velocity $\mathbf{U}$ of vehicle is constant). The key term of path following is cross-track error (y_e) which is shown to converge to zero.

We have angle of the path defined by the waypoints can be computed by:

$$\chi_p = atan2(y_{p_{k+1}} - y_{p_k}, x_{p_{k+1}} - x_{p_k}) \tag{A.62}$$

The coordinates of the kinematic vehicle in the path frame can be computed by:

$$\mathbf{e} = (x_e \quad y_e)^T = \mathbf{R}(\chi_p)(\mathbf{Q} - \mathbf{P}_{p_k}) \tag{A.63}$$

where $\mathbf{R}(\chi_p) = \begin{pmatrix} \cos(\chi_p) & \sin(\chi_p) \\ -\sin(\chi_p) & \cos(\chi_p) \end{pmatrix}$ is the rotation matrix from $\{I\}$-frame to Path-frame ($\{P\}$-frame) and $\mathbf{e}$ consists of *along-track* error x_e and *cross-track* error y_e. For straight-line path following purpose, only cross-track error is considered and is written clearly as:

$$y_e = -\sin(\chi_p)(x - x_{p_k}) + \cos(\chi_p)(y - y_{p_k}) \tag{A.64}$$

The objective of straight-line path following is to make sure that $\lim_{t\to\infty} y_e(t) = 0$. Normally, we have two classes of steering laws that ensure stabilization of

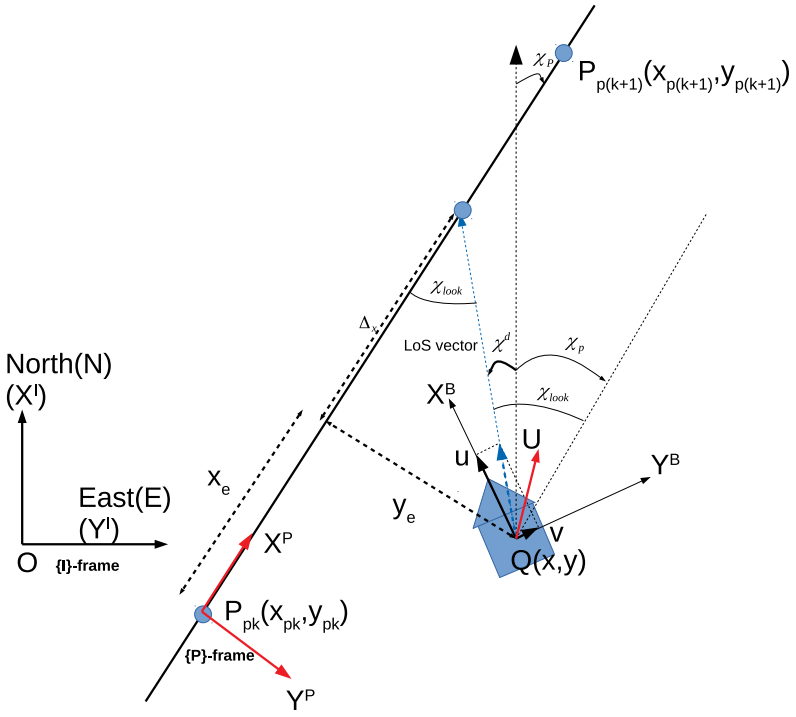


Figure A.4 – LoS principle for straight-line path-following

$y_e(t)$ converges to zero: enclosure-based steering and lookahead-based one. Only lookahead-based scheme is presented in this part. The desired course angle (χ^d) is divided into two parts:

$$\chi^d = \chi_p + \chi_{look} \tag{A.65}$$

where $\chi_{look} \triangleq \arctan(-\frac{y_e(t)}{\triangle_x})$ is a velocity-path relative angle that makes sure that the velocity directed toward a point on the path and the minus sign in definition corresponds with the sign of desired course angle. $\triangle_x > 0$ is a *lookahead distance*.

For a path formed with n-waypoints, a switching criterion is proposed as in [Morten Breivik 2009].

Curved-line path following:

A curved-path is parameterized, $\mathbf{C}(x(\zeta), y(\zeta))$. Consider an arbitrary path point $\mathbf{P}_p(\zeta)$, a path-frame ($\{P\}$-frame) is defined such that X-axis is tangent with the path at point $\mathbf{P}_p$ and Y-axis directs to right-hand side of an observer following the path (see Figure A.5). The relative angle between $\{P\}$-frame and $\{I\}$-frame is computed by:

$$\chi_p = atan2(y'_p(\zeta), x'_p(\zeta)) \tag{A.66}$$

where $y'_p(\zeta) = \frac{dy'_p}{d\zeta}, \quad x'_p(\zeta) = \frac{dx'_p}{d\zeta}$.

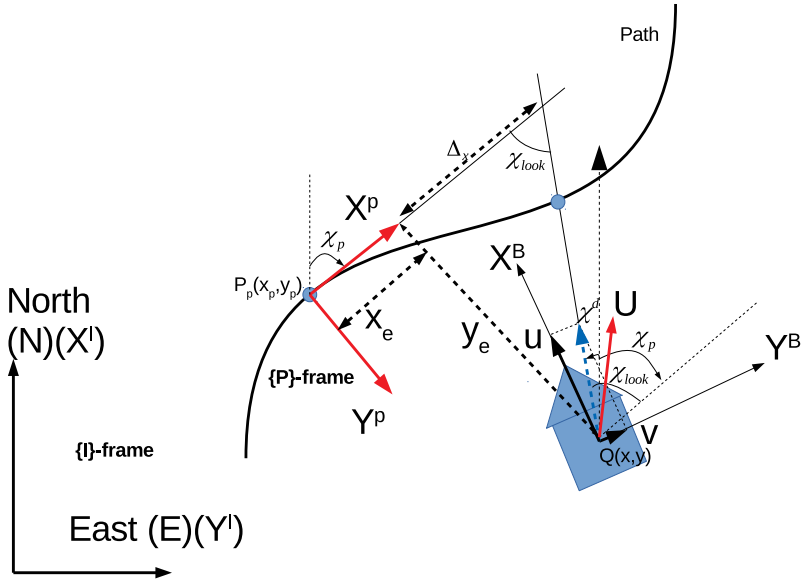


Figure A.5 – LoS principle for curved-line path-following

The coordinates of vehicle expressed in path-frame is computed by:

$$\mathbf{e}(t) = (x_e \quad y_e)^T = \mathbf{R}(\chi_p)(\mathbf{Q}(t) - \mathbf{P}_p(\zeta)) \tag{A.67}$$

where $\mathbf{R}(\chi_p) = \begin{pmatrix} \cos(\chi_p) & \sin(\chi_p) \\ -\sin(\chi_p) & \cos(\chi_p) \end{pmatrix}$ is rotation matrix from $\{I\}$-frame to $\{P\}$-frame.

The objective is $\lim_{t\to\infty} \mathbf{e}(t) = \mathbf{0}$ to reduce $\mathbf{e}(t)$ to zero.

Steering law:

$$\chi^d = \chi_p + \chi_{look} \tag{A.68}$$

where $\chi_{look} \triangleq \arctan(-\frac{y_e(t)}{\triangle_x})$.

By assigning:

$$\dot{\zeta} = \frac{\mathbf{U}\cos\chi_{look}(y_e) + \gamma x_e}{\|\mathbf{P}'_p(\zeta)\|} \tag{A.69}$$

where $\gamma > 0$ and $\|\mathbf{P}'_p(\zeta)\| = \sqrt{x'_p(\zeta)^2 + y'_p(\zeta)^2}$.

Path-following in 3D:

This section extends to curved path following in 3D. A path in 3D can be expressed by the set:

$$\mathcal{P} = \{\mathbf{P}_p \in \mathbb{R}^3 | \mathbf{P}_p = \mathbf{P}_p(\zeta), \forall\zeta \in \mathbb{R}\} \tag{A.70}$$

Defining a path-frame at arbitrary path point $\mathbf{P}_p(\zeta)$. Two consecutive rotations can be deployed for rotating $\{NED\}$-frame to path-frame ($\{P\}$-frame). The first

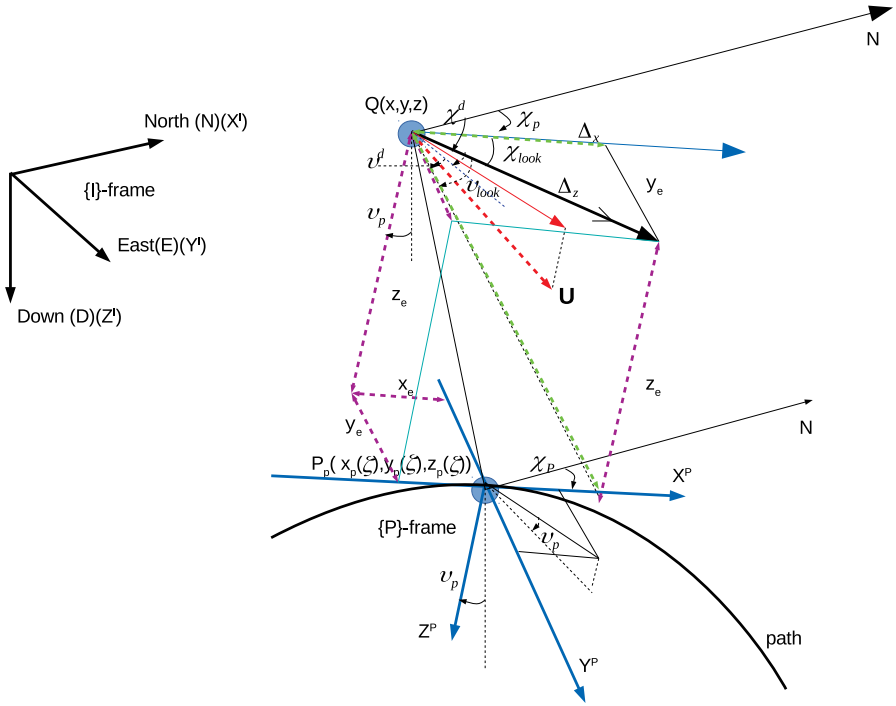


Figure A.6 – LoS principle for curved-line path-following in 3D with $\mu = 1$

is about Z-axis:

$$\chi_p(\zeta) = atan2(y'_p(\zeta), x'_p(\zeta)) \tag{A.71}$$

where $y'_p(\zeta)$ and $x'_p(\zeta)$ are derivative of positions w.r.t parameter. This rotation can be represented by a rotation matrix as:

$$\mathbf{R}(\chi_p) = \begin{pmatrix} \cos\chi_p & \sin\chi_p & 0 \\ -\sin\chi_p & \cos\chi_p & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{A.72}$$

The second one is about Y-axis:

$$\upsilon_p(\zeta) = atan2(-z'_p(\zeta), \sqrt{x'_p(\zeta)^2 + y'_p(\zeta)^2}) \tag{A.73}$$

This rotation can be represented by a rotation matrix as:

$$\mathbf{R}(\upsilon_p) = \begin{pmatrix} \cos\upsilon_p & 0 & \sin\upsilon_p \\ 0 & 1 & 0 \\ -\sin\upsilon_p & 0 & \cos\upsilon_p \end{pmatrix} \tag{A.74}$$

Errors (along-track, cross-track, and vertical-track), coordinate systems expressed in path-frame, is computed by:

$$\mathbf{e}(t) = [x_e \quad y_e \quad z_e]^T = \mathbf{R}(\chi_p, \upsilon_p)(\mathbf{Q} - \mathbf{P}_p(\zeta)) \tag{A.75}$$

where $\mathbf{R}(\chi_p, \upsilon_p) = \mathbf{R}(\upsilon_p)\mathbf{R}(\chi_p)$.

The objective is to force $\lim_{t\to\infty} \mathbf{e}(t) = \mathbf{0}$.

The steering laws for XY-plane:

$$\chi_{look}(y_e) = \arctan(-\frac{y_e(t)}{\triangle_x}) \tag{A.76}$$

where $\triangle_x$ is a chosen variable and called looking-ahead distance.

The steering laws for XZ-plane:

$$\upsilon_{look}(z_e) = \arctan(\frac{z_e(t)}{\triangle_z}) = \arctan(\frac{z_e(t)}{\mu\sqrt{y_e^2 + \triangle_x^2}}) \tag{A.77}$$

where $\mu > 0$.

The derivative of parameterize variable of path:

$$\dot{\zeta} = \frac{\mathbf{U}cos(\chi_{look}(y_e))cos(\upsilon_{look}(z_e)) + \gamma x_e(t)}{\|\mathbf{P}'_p(\zeta)\|} \tag{A.78}$$

where $\gamma > 0$ and $\|\mathbf{P}'_p(\zeta)\| = \sqrt{x'_p(\zeta)^2 + y'_p(\zeta)^2 + z'_p(\zeta)^2}$.

Finally, steering assignments:

$$\chi^d(\chi_p, \chi_{look}, \upsilon_p, \upsilon_{look}) = atan2(f, g) \tag{A.79}$$

$$\upsilon^d(\chi_p, \chi_{look}, \upsilon_p, \upsilon_{look}) = \arcsin(\sin\upsilon_{look}\cos\chi_{look}\cos\upsilon_{look} + \cos\upsilon_p\sin\upsilon_{look}) \tag{A.80}$$

where

$$f = \cos\chi_p\sin\chi_{look}\cos\upsilon_{look} - \sin\chi_p\sin\upsilon_p\sin\upsilon_{look} + \sin\chi_p\cos\upsilon_p\cos\chi_{look}\cos\upsilon_{look} \tag{A.81}$$

and

$$g = -\sin\chi_p\sin\chi_{look}\cos\upsilon_{look} - \cos\chi_p\sin\upsilon_p\sin\upsilon_{look} + \cos\chi_p\cos\upsilon_p\cos\chi_{look}\cos\upsilon_{look} \tag{A.82}$$

A.5.2 Virtual frame tracking

Serret-Frenet frame for path-following: This section, based on [Taylor 1972] and [Martins-Encarnação 2002], presents Serret-Frenet frame for path-following problem, denoted as $\{SF\}$, it departs from conventional Serret-Frenet frame. Consider a parameterized path in a fixed-frame as follows:

$$\mathbf{P}_p(\zeta) = [x_p(\zeta) \quad y_p(\zeta) \quad z_p(\zeta)]^T \tag{A.83}$$

The length of a smooth arc from $\zeta = a$ to $\zeta = b$ is given by:

$$L = \int_a^b \sqrt{(\frac{dx}{d\zeta})^2 + (\frac{dy}{d\zeta})^2 + (\frac{dz}{d\zeta})^2}d\omega \tag{A.84}$$

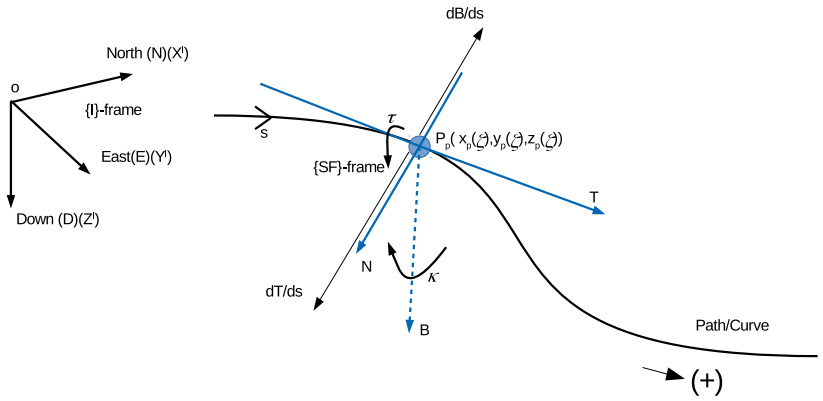


Figure A.7 – Serret-Frenet frame

Arc-length is defined as the length from $\zeta = a$ to a variable point on that arc (replacing the upper limit b in (A.84) and differentiating derives):

$$s = \int_a^{\zeta} \sqrt{(\frac{dx}{d\zeta})^2 + (\frac{dy}{d\zeta})^2 + (\frac{dz}{d\zeta})^2} d\zeta \tag{A.85}$$

$$\frac{ds}{d\zeta} = \pm\sqrt{(\frac{dx}{d\zeta})^2 + (\frac{dy}{d\zeta})^2 + (\frac{dz}{d\zeta})^2} \tag{A.86}$$

Note that the minus sign if s and ζ vary in different directions.

So, we have:

$$ds^2 = dx^2 + dy^2 + dz^2 \tag{A.87}$$

Along the arc, choose a defined direction as the positive direction. At arbitrary point of the path, define a vector $\mathbf{T}$ as an unit-length vector along the tangent to the curve in the positive direction.

It is easily to prove that if $\mathbf{F}$ is a vector of constant length, then $\mathbf{F}.\frac{d\mathbf{F}}{ds} = 0$, and thus conclude that $\frac{d\mathbf{F}}{ds}$ is perpendicular to $\mathbf{F}$ unless $\frac{d\mathbf{F}}{ds}(\frac{d(\mathbf{F}.\mathbf{F})}{ds} = 2\mathbf{F}\frac{d\mathbf{F}}{ds} = 0$ since it is a vector of constant length).

From preceding fact, we see that vector $\frac{d\mathbf{T}}{ds}$ is perpendicular to vector $\mathbf{T}$ and a unit vector in its direction is called the *principal normal* to C at the point in question.

Now, we derive the definition of Serret-Frenet frame which is used in the path-following problem because the principal normal is not suitable. In fact, considering in the case of planar curves, the principal normal lies in the plane of the curve and points towards the concave side of the curve. Thus, if the curve is a sinusoid, the direction of the principal normal would be discontinuous, jumping by 180^0 at the inflection points of the curve. So, for overcoming this difficulty, a normal, denoted

by $\mathbf{N}$, is defined that "always points to the right side of the curve" and can be computed as:

$$\frac{d\mathbf{T}}{ds} = \kappa\mathbf{N} \tag{A.88}$$

where κ is called the *curvature* of C and is positive if the curve turns right and negative otherwise.

In summary, in path-following problem, $\mathbf{N}$ is a vector parallel to the horizontal plane, perpendicular to the tangent vector, and pointing to the right of an observer travelling along the positive direction of the path. The plane defined by $\mathbf{T}$ and $\mathbf{N}$ at a given point $\mathbf{P}$ is called as the osculating plane of C at $\mathbf{P}$.

Now,

$$\mathbf{B} = \mathbf{T} \otimes \mathbf{N} \tag{A.89}$$

denotes the *binormal* vector of C at point $\mathbf{P}$. We see that $\mathbf{T}$, $\mathbf{N}$, and $\mathbf{B}$ are mutually perpendicular unit vectors forming a right-handed system and the frame formed by these vectors is called as the Serret-Frame and will be denoted $\{SF\}$. Note that it is different from conventional Serret-Frenet frame in which the vector $\mathbf{N}$ is the principal normal.

To define *torsion*, considering $\frac{d\mathbf{B}}{ds}$, From Equation (A.89),

$$\frac{d\mathbf{B}}{ds} = \mathbf{T} \otimes \frac{d\mathbf{N}}{ds} + \frac{d\mathbf{T}}{ds} \otimes \mathbf{N} \tag{A.90}$$

From Equation (A.88), we have:

$$\frac{d\mathbf{T}}{ds} \otimes \mathbf{N} = \kappa\mathbf{N} \otimes \mathbf{N} = 0 \tag{A.91}$$

Therefore,

$$\frac{d\mathbf{B}}{ds} = \mathbf{T} \otimes \frac{d\mathbf{N}}{ds} \tag{A.92}$$

As aforementioned fact, since $\mathbf{B}$ is an unit vector,

$$\mathbf{B}\frac{d\mathbf{B}}{ds} = 0 \tag{A.93}$$

If $\frac{d\mathbf{B}}{ds} \neq 0$, Equations (A.92) and (A.93) show that $\frac{d\mathbf{B}}{ds}$ is perpendicular to both $\mathbf{B}$ and $\mathbf{T}$ and is therefore a multiple of $\mathbf{N}$. So, it can be written as:

$$\frac{d\mathbf{B}}{ds} = -\tau\mathbf{N} \tag{A.94}$$

The scalar quantity τ is called the torsion of C at the point under consideration. If $\frac{d\mathbf{B}}{ds} = 0$, τ is defined as zero such that (A.94) still holds. In the case of planar curves, $\mathbf{B}$ is a constant and unit vector perpendicular to the plane and $\tau = 0$ at all points. A curve which does not lie in a single plane is called a twisted curve and its torsion computes the amount by which the curve is twisted.

We investigate the derivative of $\mathbf{N}$ vector:

$$\frac{d\mathbf{N}}{ds} = -\tau\mathbf{B} - \kappa\mathbf{T} \tag{A.95}$$

It is easy to prove the following useful relations:

$$\frac{d\psi}{d\zeta} = \kappa \frac{ds}{d\zeta} \tag{A.96}$$

$$\frac{d\phi}{d\zeta} = \tau \frac{ds}{d\zeta} \tag{A.97}$$

where ψ is the rotation angle of the $\{SF\}$ frame about $\mathbf{B}$ and ϕ is the rotation angle of the one about $\mathbf{T}$.

How to compute κ and τ?.

$$\kappa = \frac{\|\mathbf{P}'_p \otimes \mathbf{P}''_p\|}{\|\mathbf{P}'_p\|^3} \tag{A.98}$$

$$\tau = \frac{(\mathbf{P}'_p \otimes \mathbf{P}''_p).\mathbf{P}'''_p}{\|\mathbf{P}_p \otimes \mathbf{P}''_p\|^2} \tag{A.99}$$

where primes denote differentiation w.r.t ζ.

Path-following:

Consider a reference path, $\mathbf{P}(\zeta)$ with parameters s, k, and τ and consider a $\{SF\}$ frame associated to a point p moving along that curve with fixed speed $\dot{s}$.

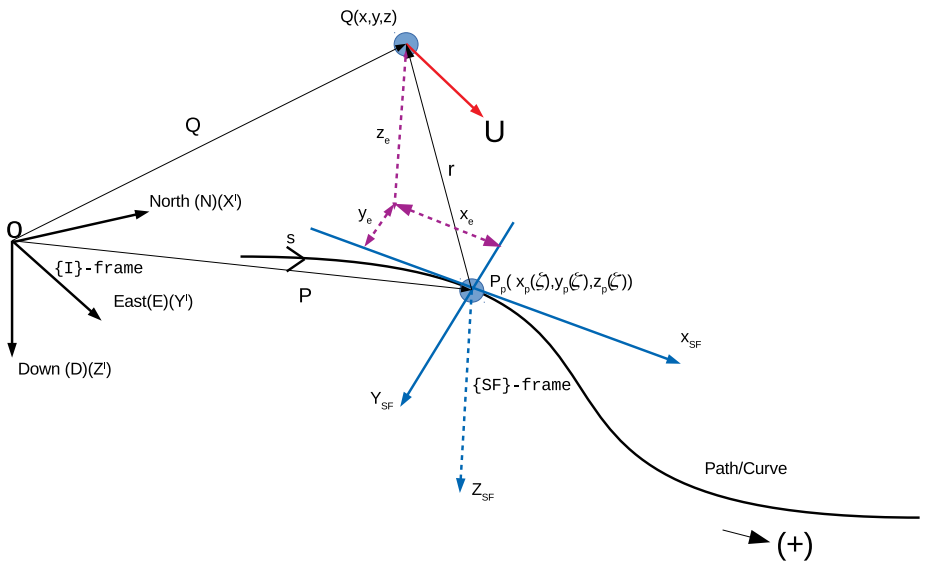


Figure A.8 – Path-following with Serret-Frenet frame

The linear velocity of $\{SF\}$ measured in $\{I\}$ and expressed in $\{SF\}$ is given by:

$${}^{SF}\boldsymbol{\nu}_{SF} = [\dot{s} \quad 0 \quad 0]^T \tag{A.100}$$

The angular velocity of $\{SF\}$ measured in $\{I\}$ and expressed in $\{SF\}$ is given by:

$$^{SF}\boldsymbol{\omega}_{SF} = [\tau\dot{s} \quad 0 \quad \kappa\dot{s}]^T \tag{A.101}$$

Using *roll-pitch-yaw* convention, we can express linear and angular velocity in NED-frame as:

$$^{I}\boldsymbol{\nu}_{SF} = \begin{pmatrix} \dot{x}_{SF} \\ \dot{y}_{SF} \\ \dot{z}_{SF} \end{pmatrix} = ^{I}\mathbf{R}_{SF}(\phi_{SF}, \theta_{SF}, \psi_{SF})\,^{SF}\boldsymbol{\nu}_{SF} \tag{A.102}$$

$$^{I}\boldsymbol{\omega}_{SF} = \begin{pmatrix} \dot{\phi}_{SF} \\ \dot{\theta}_{SF} \\ \dot{\psi}_{SF} \end{pmatrix} = ^{I}\mathbf{Q}_{SF}(\phi_{SF}, \theta_{SF}, \psi_{SF})\,^{SF}\boldsymbol{\omega}_{SF} \tag{A.103}$$

where $^{I}\mathbf{R}_{SF}$ and $^{I}\mathbf{Q}_{SF}$ are rotation matrix from $\{SF\}$ frame to I frame as follows:

$$^{I}\mathbf{R}_{SF} = \begin{pmatrix} c(\psi_{SF})c(\theta_{SF}) & -s(\psi_{SF})c(\phi_{SF})+s(\phi_{SF})s(\theta_{SF})c(\psi_{SF}) & s(\psi_{SF})s(\phi_{SF})+s(\theta_{SF})c(\psi_{SF})c(\phi_{SF}) \\ s(\psi_{SF})c(\theta_{SF}) & c(\psi_{SF})c(\phi_{SF})+s(\phi_{SF})s(\theta_{SF})s(\psi_{SF}) & -c(\psi_{SF})s(\phi_{SF})+s(\theta_{SF})s(\psi_{SF})c(\phi_{SF}) \\ -s(\theta_{SF}) & s(\phi_{SF})c(\theta_{SF}) & c(\phi_{SF})c(\theta_{SF}) \end{pmatrix} \tag{A.104}$$

$$^{I}\mathbf{Q}_{SF} = \begin{pmatrix} 1 & s(\phi_{SF})t(\theta_{SF}) & c(\phi_{SF})t(\theta_{SF}) \\ 0 & c(\phi_{SF}) & -s(\phi_{SF}) \\ 0 & \frac{s(\phi_{SF})}{c(\theta_{SF})} & \frac{c(\phi_{SF})}{c(\theta_{SF})} \end{pmatrix} \tag{A.105}$$

where $c() = cos()$, $s() = sin()$, and $t() = tan()$.

Therefore, the kinematic model of $\{SF\}$ frame is expressed in compact form as:

$$\begin{pmatrix} \dot{x}_{SF} \\ \dot{y}_{SF} \\ \dot{z}_{SF} \\ \dot{\phi}_{SF} \\ \dot{\theta}_{SF} \\ \dot{\psi}_{SF} \end{pmatrix} = \begin{pmatrix} \dot{s}\cos(\psi_{SF})\cos(\theta_{SF}) \\ \dot{s}\sin(\psi_{SF})\sin(\phi_{SF}) \\ \dot{s} - \sin(\theta_{SF}) \\ \tau\dot{s} + \kappa\dot{s}\cos(\phi_{SF})\tan(\theta_{SF}) \\ -\kappa\dot{s}\sin(\phi_{SF}) \\ \kappa\dot{s}\frac{\cos(\phi_{SF})}{\cos(\theta_{SF})} \end{pmatrix} \tag{A.106}$$

The underlying idea of using $\{SF\}$ frame kinematic for path-following problem is to coincide Body frame of vehicle and $\{SF\}$ frame at a point of the reference curve.

The coordinates of $\mathbf{Q}$ expressed in I-frame is: $^{I}\mathbf{Q} = [x \quad y \quad z]^T$ and in $\{SF\}$-frame is: $^{SF}\mathbf{Q} = [x_e \quad y_e \quad z_e]^T$ (see Figure A.8). Then, the relative velocity between the wind (Body-frame if wind is neglected) and the Serret-Frenet frames is $\frac{d^{SF}\mathbf{Q}}{dt} = [\dot{x_e} \quad \dot{y_e} \quad \dot{z_e}]^T$. Now, we find the evolution of these errors?.

Following Figure A.8, it is straightforward to compute the linear velocity of $\mathbf{Q}$ in I-frame as [Lapierre 2006c]:

$$^{I}(\frac{d\mathbf{Q}}{dt}) = ^{I}(\frac{d\mathbf{P}}{dt}) + ^{SF}\mathbf{R}_I^T\,^{SF}(\frac{d\mathbf{r}}{dt}) + ^{SF}\mathbf{R}_I^T(\boldsymbol{\omega} \otimes ^{SF}\mathbf{r}) \tag{A.107}$$

where ${}^{SF}\mathbf{R}_I = {}^I\mathbf{R}_{SF}^T$ is the rotation matrix from $\{I\}$-frame to $\{SF\}$-frame, ω is the angular velocity of $\{SF\}$ frame with respect to $\{I\}$ frame.

Multiplying both sides of Equation (A.107) with $\mathbf{R}$ yields:

$$ {}^{SF}\mathbf{R}_I \, {}^I(\frac{d\mathbf{Q}}{dt}) = {}^{SF}(\frac{d\mathbf{P}}{dt}) + {}^{SF}(\frac{d\mathbf{r}}{dt}) + (\boldsymbol{\omega} \otimes {}^{SF}\mathbf{r}) \tag{A.108} $$

This is expanded as:

$$ {}^{SF}\mathbf{R}_I(\phi_{SF}, \theta_{SF}, \psi_{SF}) \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = \begin{pmatrix} \dot{s} \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{z}_e \end{pmatrix} + \begin{pmatrix} \dot{\phi}_{SF} \\ \dot{\theta}_{SF} \\ \dot{\psi}_{SF} \end{pmatrix} \otimes \begin{pmatrix} x_e \\ y_e \\ z_e \end{pmatrix} \tag{A.109} $$

On the other hand, the kinematic model of the vehicle is given by:

$$ \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = {}^I\mathbf{R}_B(\phi, \theta, \psi) \begin{pmatrix} u \\ v \\ w \end{pmatrix} \tag{A.110} $$

where ${}^I\mathbf{R}_B$ is the rotation matrix from $\{B\}$-frame (Body-frame) to $\{I\}$-frame.

By introducing $\phi_e = \phi - \phi_{SF}$, $\theta_e = \theta - \theta_{SF}$, and $\psi_e = \psi - \psi_{SF}$, from above Equations, we can get the "kinematic model" of the vehicle expressed in $\{SF\}$-frame as:

$$ \begin{pmatrix} \dot{s} \\ \dot{x}_e \\ \dot{y}_e \\ \dot{z}_e \\ \dot{\phi}_e \\ \dot{\theta}_e \\ \dot{\psi}_e \end{pmatrix} = f(s, u, v, w, \phi_e, \theta_e, \psi_e, x_e, y_e, z_e, \kappa, \tau) \tag{A.111} $$

The control objective is to force errors to converge. Nonlinear control methods, i.e, back-stepping, gain-scheduling, MPC, can be applied to solve path-following problem.

The controller of path-following problem is as follows:

$$ F_u = -m_{22}vr + m_{33}wq + d_{11}u + m_{11}(K_{pu}e_u + K_{iu}\int_0^t e_u dt + K_{du}\frac{de_u}{dt}) \tag{A.112} $$

$$ F_p = (m_{33} - m_{22})vw + (m_{66} - m_{55})qr + d_{44}p + m_{44}(K_{pp}e_\phi + K_{ip}\int_0^t e_\phi dt + K_{du}\frac{de_\phi}{dt}) \tag{A.113} $$

$$ F_q = (m_{11} - m_{33})uw + (m_{44} - m_{66})pr + d_{55}q + m_{55}(K_{pq}e_\theta + K_{iq}\int_0^t e_\theta dt + K_{dq}\frac{de_\theta}{dt}) \tag{A.114} $$

$$ F_r = (m_{22} - m_{11})uv + (m_{55} - m_{44})pq + d_{66}r + m_{66}(K_{pr}e_\psi + K_{ir}\int_0^t e_\psi dt + K_{dr}\frac{de_\psi}{dt}) \tag{A.115} $$

where m_{ii}, d_{ii} are dynamic parameters of the robot. K_* is control gain.

Appendix B

Appendix

B.1 Dynamic model of marine robots

Following [Fossen 2011], rigid-body inertia matrix $\mathbf{M}_{RB}$ is unique and has the form:

$$\mathbf{M}_{RB} = \begin{pmatrix} m & 0 & 0 & 0 & mz_g & -my_g \\ 0 & m & 0 & -mz_g & 0 & mx_g \\ 0 & 0 & m & my_g & -mx_g & 0 \\ 0 & -mz_g & my_g & I_x & -I_{xy} & -I_{xz} \\ mz_g & 0 & -mx_g & -I_{yx} & I_y & -I_{yz} \\ -my_g & mx_g & 0 & -I_{zx} & -I_{zy} & I_z \end{pmatrix} \tag{B.1}$$

where m is the mass of the robot, $[x_g \quad y_g \quad z_g]^T$ is the coordinates of Center of Gravity (**CG**) expressed in $\{B\}$, I_x, I_y, I_z are the moments of inertia about x_B, y_B, z_B axes respectively; $I_{xy} = I_{yx}$, $I_{xz} = I_{zx}$, $I_{yz} = I_{zy}$ are the products of inertia.

The added inertia matrix is given by:

$$\mathbf{M}_A = -\begin{pmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{w}} & X_{\dot{p}} & X_{\dot{q}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{w}} & Y_{\dot{p}} & Y_{\dot{q}} & Y_{\dot{r}} \\ Z_{\dot{u}} & Z_{\dot{v}} & Z_{\dot{w}} & Z_{\dot{p}} & Z_{\dot{q}} & Z_{\dot{r}} \\ K_{\dot{u}} & K_{\dot{v}} & K_{\dot{w}} & K_{\dot{p}} & K_{\dot{q}} & K_{\dot{r}} \\ M_{\dot{u}} & M_{\dot{v}} & M_{\dot{w}} & M_{\dot{p}} & M_{\dot{q}} & M_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{w}} & N_{\dot{p}} & N_{\dot{q}} & N_{\dot{r}} \end{pmatrix} \tag{B.2}$$

where hydrodynamic derivatives are used in this equation; for instance the hydrodynamic added mass force X along the x axis due to the acceleration $\dot{u}$ in the x direction is written as:

$$X = -X_{\dot{u}}, \quad X_{\dot{u}} := \frac{\partial X}{\partial \dot{u}} \tag{B.3}$$

Corriolis-Centripetal matrices are as follows:

$$\mathbf{C}_{RB} = \begin{pmatrix} 0 & 0 & 0 & m(y_gq+z_gr) & -m(x_gq-w) & -m(x_gr+v) \\ 0 & 0 & 0 & -m(y_gp+w) & m(z_gr+x_gp) & -m(y_gr-u) \\ 0 & 0 & 0 & -m(z_gp-v) & -m(z_gq+u) & m(x_gp+y_gq) \\ -m(y_gq+z_gr) & m(y_gp+w) & m(z_gp-v) & 0 & -I_{yz}-I_{xz}p+I_zr & I_{yz}r+I_{xy}p-I_yq \\ m(x_gq-w) & -m(z_gr+x_gp) & m(z_gq+u) & I_{yz}q+I_{xz}p-I_zr & 0 & -I_{xz}r-I_{xy}q+I_xp \\ m(x_gr+v) & m(y_gr-u) & -m(x_gp+y_gq) & -I_{yz}r-I_{xy}p+I_yq & I_{xz}r+I_{xy}q-I_xp & 0 \end{pmatrix} \tag{B.4}$$

$$\mathbf{C}_A = \begin{pmatrix} 0 & 0 & 0 & 0 & -a_3 & a_2 \\ 0 & 0 & 0 & a_3 & 0 & -a_1 \\ 0 & 0 & 0 & -a_2 & a_1 & 0 \\ 0 & -a_3 & a_2 & 0 & -b_3 & b_2 \\ a_3 & 0 & -a_1 & b_3 & 0 & -b_1 \\ -a_2 & a_1 & 0 & -b_2 & b_1 & 0 \end{pmatrix} \tag{B.5}$$

where

$$\begin{aligned} a_1 &= X_{\dot{u}}u + X_{\dot{v}}v + X_{\dot{w}}w + X_{\dot{p}}p + X_{\dot{q}}q + X_{\dot{r}}r \\ a_2 &= Y_{\dot{u}}u + Y_{\dot{v}}v + Y_{\dot{w}}w + Y_{\dot{p}}p + Y_{\dot{q}}q + Y_{\dot{r}}r \\ a_3 &= Z_{\dot{u}}u + Z_{\dot{v}}v + Z_{\dot{w}}w + Z_{\dot{p}}p + Z_{\dot{q}}q + Z_{\dot{r}}r \\ b_1 &= K_{\dot{u}}u + K_{\dot{v}}v + K_{\dot{w}}w + K_{\dot{p}}p + K_{\dot{q}}q + K_{\dot{r}}r \\ b_2 &= M_{\dot{u}}u + M_{\dot{v}}v + M_{\dot{w}}w + M_{\dot{p}}p + M_{\dot{q}}q + M_{\dot{r}}r \\ b_3 &= N_{\dot{u}}u + N_{\dot{v}}v + N_{\dot{w}}w + N_{\dot{p}}p + N_{\dot{q}}q + N_{\dot{r}}r \end{aligned} \tag{B.6}$$

Damping matrices have the forms:

$$\mathbf{D}_l = -\begin{pmatrix} X_u & X_v & X_w & X_p & X_q & X_r \\ Y_u & Y_v & Y_w & Y_p & Y_q & Y_r \\ Z_u & Z_v & Z_r & Z_p & Z_q & Z_r \\ K_u & K_v & K_w & K_p & K_q & K_r \\ M_u & M_v & M_w & M_p & M_q & M_r \\ N_u & N_v & N_w & N_p & N_q & N_r \end{pmatrix} \tag{B.7}$$

$$\mathbf{D}_n = \begin{pmatrix} X_{|u|u}|u| & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{|v|v}|v|+Y_{|r|v}|r| & 0 & 0 & 0 & Y_{|v|r}|v|+Y_{|r|r}|r| \\ 0 & 0 & Z_{|w|w}|w| & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{|p|p}|p| & 0 & 0 \\ 0 & 0 & 0 & 0 & M_{|q|q}|q| & 0 \\ 0 & N_{|v|v}|v|+N_{|r|v}|r| & 0 & 0 & 0 & N_{|v|r}|v|+N_{|r|r}|r| \end{pmatrix} \tag{B.8}$$

where X_u is linear damping force along x-axis and $X_{|u|u}$ are nonlinear damping coefficient along x-axis. Readers can refer to [Fossen 2011] for more details.

Hydrostatics of a marine vehicle is given by:

$$\mathbf{g}(\eta) = \begin{pmatrix} (W-B)\sin(\theta) \\ -(W-B)\cos(\theta)\sin(\phi) \\ -(W-B)\cos(\theta)\cos(\phi) \\ -(y_gW - y_bB)\cos(\theta)\cos(\phi) + (z_gW - z_bB)\cos(\theta)\sin(\phi) \\ (z_gW - z_bB)\sin(\theta) + (x_gW - x_bB)\cos(\theta)\cos(\phi) \\ -(x_gW - x_bB)\cos(\theta)\sin(\phi) - (y_gW - y_bB)\sin(\theta) \end{pmatrix} \tag{B.9}$$

B.2 6DOFs dynamics model of AUVs

This section shows the simplified model of 6-DOFs AUVs. We assume that Center of Gravity(**CG**) coincides with Center of Origin of body-frame (**CO**). The

robot has three planes of symmetry, no wind and wave, and the robot is designed with buoyancy natually. There are no couplings in the matrices $\mathbf{M}$, $\mathbf{D}_l$, and $\mathbf{D}_n$. All matrices in dynamic model are described as follows:

$$\mathbf{M}_{RB} = \begin{pmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_x & 0 & 0 \\ 0 & 0 & 0 & 0 & I_y & 0 \\ 0 & 0 & 0 & 0 & 0 & I_z \end{pmatrix} \tag{B.10}$$

$$\mathbf{M}_A = - \begin{pmatrix} X_{\dot{u}} & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{\dot{v}} & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_{\dot{w}} & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{\dot{p}} & 0 & 0 \\ 0 & 0 & 0 & 0 & M_{\dot{q}} & 0 \\ 0 & 0 & 0 & 0 & 0 & N_{\dot{r}} \end{pmatrix} \tag{B.11}$$

Therefore, $\mathbf{M}$ matrix is written by:

$$\mathbf{M} = \begin{pmatrix} m - X_{\dot{u}} & 0 & 0 & 0 & 0 & 0 \\ 0 & m - Y_{\dot{v}} & 0 & 0 & 0 & 0 \\ 0 & 0 & m - Z_{\dot{w}} & 0 & 0 & 0 \\ 0 & 0 & 0 & I_x - K_{\dot{p}} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_y - M_{\dot{q}} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_z - N_{\dot{r}} \end{pmatrix} \tag{B.12}$$

or

$$\mathbf{M} = \begin{pmatrix} m_{11} & 0 & 0 & 0 & 0 & 0 \\ 0 & m_{22} & 0 & 0 & 0 & 0 \\ 0 & 0 & m_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & m_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & m_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & m_{66} \end{pmatrix} \tag{B.13}$$

where $m_{11} = m - X_{\dot{u}}, m_{22} = m - Y_{\dot{v}}, m_{33} = m - Z_{\dot{w}}, m_{44} = I_x - K_{\dot{p}}, m_{55} = I_y - M_{\dot{q}}$ and $m_{66} = I_z - N_{\dot{r}}$

The Corriloris-Centriputal matrices:

$$\mathbf{C}_{RB} = \begin{pmatrix} 0 & 0 & 0 & 0 & mw & -mv \\ 0 & 0 & 0 & -mw & 0 & mu \\ 0 & 0 & 0 & mv & -mu & 0 \\ 0 & mw & -mv & 0 & I_z r & -I_y q \\ -mw & 0 & mu & -I_z r & 0 & I_x p \\ mv & -mu & 0 & I_y q & -I_x p & 0 \end{pmatrix} \tag{B.14}$$

$$\mathbf{C}_A = \begin{pmatrix} 0 & 0 & 0 & 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v \\ 0 & 0 & 0 & Z_{\dot{w}}w & 0 & -X_{\dot{u}}u \\ 0 & 0 & 0 & -Y_{\dot{v}}v & X_{\dot{u}}u & 0 \\ 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v & 0 & -N_{\dot{r}}r & M_{\dot{q}}q \\ Z_{\dot{w}}w & 0 & -X_{\dot{u}}u & N_{\dot{r}}r & 0 & -K_{\dot{p}}p \\ -Y_{\dot{v}}v & X_{\dot{u}}u & 0 & -M_{\dot{q}}q & K_{\dot{p}}p & 0 \end{pmatrix} \tag{B.15}$$

Therefore, the matrix $\mathbf{C}$ is written by:

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & 0 & 0 & (m-Z_{\dot{w}})w & -(m-Y_{\dot{v}})v \\ 0 & 0 & 0 & -(m-Z_{\dot{w}})w & 0 & (m-X_{\dot{u}})u \\ 0 & 0 & 0 & (m-Y_{\dot{v}})v & -(m-X_{\dot{u}})u & 0 \\ 0 & (m-Z_{\dot{w}})w & -(m-Y_{\dot{v}})v & 0 & (I_z-N_{\dot{r}})r & -(I_y-M_{\dot{q}})q \\ -(m-Z_{\dot{w}})w & 0 & (m-X_{\dot{u}})u & -(I_z-N_{\dot{r}})r & 0 & (I_x-K_{\dot{p}})p \\ (m-Y_{\dot{v}})v & -(m-X_{\dot{u}})u & 0 & (I_y-M_{\dot{q}})q & -(I_x-K_{\dot{p}})p & 0 \end{pmatrix} \tag{B.16}$$

or

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & 0 & 0 & m_{33}w & -m_{22}v \\ 0 & 0 & 0 & -m_{33}w & 0 & m_{11}u \\ 0 & 0 & 0 & m_{22}v & -m_{11}u & 0 \\ 0 & m_{33}w & -m_{22}v & 0 & m_{66}r & -m_{44}q \\ -m_{33}w & 0 & m_{11}u & -m_{66}r & 0 & m_{44}p \\ m_{22}v & -m_{11}u & 0 & m_{44}q & -m_{44}p & 0 \end{pmatrix} \tag{B.17}$$

Damping matrices:

$$\mathbf{D}_l = -\begin{pmatrix} X_u & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_v & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_r & 0 & 0 & 0 \\ 0 & 0 & 0 & K_p & 0 & 0 \\ 0 & 0 & 0 & 0 & M_q & 0 \\ 0 & 0 & 0 & 0 & 0 & N_r \end{pmatrix} \tag{B.18}$$

$$\mathbf{D}_n = \begin{pmatrix} X_{|u|u}|u| & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{|v|v}|v| & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_{|w|w}|w| & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{|p|p}|p| & 0 & 0 \\ 0 & 0 & 0 & 0 & M_{|q|q}|q| & 0 \\ 0 & 0 & 0 & 0 & 0 & N_{|r|r}|r| \end{pmatrix} \tag{B.19}$$

The damping matrix $\mathbf{D}$ is written by:

$$\mathbf{D} = \begin{pmatrix} X_{|u|u}|u| - X_u & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{|v|v}|v| - Y_v & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_{|w|w}|w| - Z_r & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{|p|p}|p| - K_p & 0 & 0 \\ 0 & 0 & 0 & 0 & M_{|q|q}|q| - M_q & 0 \\ 0 & 0 & 0 & 0 & 0 & N_{|r|r}|r| - N_r \end{pmatrix} \tag{B.20}$$

or

$$\mathbf{D} = \begin{pmatrix} d_{11} & 0 & 0 & 0 & 0 & 0 \\ 0 & d_{22} & 0 & 0 & 0 & 0 \\ 0 & 0 & d_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & d_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & d_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & d_{66} \end{pmatrix} \tag{B.21}$$

where $d_{11} = X_{|u|u}|u| - X_u, d_{22} = Y_{|v|v}|v| - Y_v, d_{33} = Z_{|w|w}|w| - Z_r, d_{44} = K_{|p|p}|p| - K_p, d_{55} = M_{|q|q}|q| - M_q$ and $d_{66} = N_{|r|r}|r| - N_r$.

The restoring forces and moments:

$$\mathbf{g}(\eta) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -\overline{BG}_y W\cos(\theta)\cos(\phi) + \overline{BG}_z W\cos(\theta)\sin(\phi) \\ \overline{BG}_z W\sin(\theta) + \overline{BG}_x W\cos(\theta)\cos(\phi) \\ -\overline{BG}_x W\cos(\theta)\sin(\phi) - \overline{BG}_y W\sin(\theta) \end{pmatrix} = \begin{pmatrix} X_G(\eta) \\ Y_G(\eta) \\ Z_G(\eta) \\ K_G(\eta) \\ M_G(\eta) \\ N_G(\eta) \end{pmatrix} \tag{B.22}$$

The simplified dynamic model of AUVs is written as:

$$F_u = (m - X_{\dot{u}})\dot{u} + (m - Z_{\dot{w}})wq - (m - Y_{\dot{v}})vr + (X_{|u|u}|u| - X_u)u + X_G(\eta) \tag{B.23}$$

$$F_v = (m - Y_{\dot{v}})\dot{v} - (m - Z_{\dot{w}})wp + (m - X_{\dot{u}})ur + (Y_{|v|v}|v| - Y_v)v + Y_G(\eta) \tag{B.24}$$

$$F_w = (m - Z_{\dot{w}})\dot{w} + (m - Y_{\dot{v}})vp - (m - X_{\dot{u}})uq + (Z_{|w|w}|w| - Z_w)w + Z_G(\eta) \tag{B.25}$$

$$\Gamma_p = (I_x - K_{\dot{p}})\dot{p} + (Y_{\dot{v}} - Z_{\dot{w}})vw + (I_z - I_y - N_{\dot{r}} + M_{\dot{q}})qr + (K_{|p|p}|p| - K_p)p + K_G(\eta) \tag{B.26}$$

$$\Gamma_q = (I_y - M_{\dot{q}})\dot{q} + (Z_{\dot{w}} - X_{\dot{u}})uw + (I_x - I_z + N_{\dot{r}} - K_{\dot{p}})pr + (M_{|q|q}|q| - M_q)q + M_G(\eta) \tag{B.27}$$

$$\Gamma_r = (I_z - N_{\dot{r}})\dot{r} + (X_{\dot{u}} - Y_{\dot{v}})uv + (I_y - I_x + K_{\dot{p}} - M_{\dot{q}})pq + (N_{|r|r}|r| - N_r)r + N_G(\eta) \tag{B.28}$$

or

$$F_u = m_{11}\dot{u} + m_{33}wq - m_{22}vr + d_{11}u + X_G(\eta) \tag{B.29}$$
$$F_v = m_{22}\dot{v} - m_{33}wp + m_{11}ur + d_{22}v + Y_G(\eta) \tag{B.30}$$
$$F_w = m_{33}\dot{w} + m_{22}vp - m_{11}uq + d_{33}w + Z_G(\eta) \tag{B.31}$$
$$\Gamma_p = m_{44}\dot{p} + (m_{33} - m_{22})vw + (m_{66} - m_{55})qr + d_{44}p + K_G(\eta) \tag{B.32}$$
$$\Gamma_q = m_{55}\dot{q} + (m_{11} - m_{33})uw + (m_{44} - m_{66})pr + d_{55}q + M_G(\eta) \tag{B.33}$$
$$\Gamma_r = m_{66}\dot{r} + (m_{22} - m_{11})uv + (m_{55} - m_{44})pq + d_{66}r + N_G(\eta) \tag{B.34}$$

In some cases, it is reasonable to ignore the roll model and nonlinear damping terms, the kinematic and dynamic models are simplified more and written as:

1. Kinematic model:

$$\dot{x} = ucos(\psi)cos(\theta) - vsin(\psi) + wsin(\theta)cos(\psi) \tag{B.35}$$
$$\dot{y} = usin(\psi)cos(\theta) + vcos(\psi) + wsin(\theta)sin(\psi) \tag{B.36}$$
$$\dot{z} = -usin(\theta) + w\cos(\theta) \tag{B.37}$$
$$\dot{\theta} = q \tag{B.38}$$
$$\dot{\psi} = \frac{r}{cos(\theta)} \tag{B.39}$$

2. Dynamic model:

$$\dot{u} = \frac{m_{22}}{m_{11}}vr - \frac{m_{33}}{m_{11}}wq - \frac{d_{11}}{m_{11}} + \frac{1}{m_{11}}X \tag{B.40}$$
$$\dot{v} = -\frac{m_{11}}{m_{22}}ur - \frac{d_{22}}{m_{22}}v \tag{B.41}$$
$$\dot{w} = \frac{m_{11}}{m_{33}}uq - \frac{d_{33}}{m_{33}}w \tag{B.42}$$
$$\dot{q} = \frac{m_{33} - m_{11}}{m_{55}}uw - \frac{d_{55}}{m_{55}}q - \frac{M_G}{m_{55}} + \frac{1}{m_{55}}M \tag{B.43}$$
$$\dot{r} = \frac{m_{11} - m_{22}}{m_{66}}uv - \frac{d_{66}}{m_{66}}r + \frac{1}{m_{66}}N \tag{B.44}$$

Appendix C

Appendix

C.1 IMU calibration

This section briefly presents the basic theories of Inertial Measurement Unit (IMU) calibration and results of robot's IMU calibration. The attitude estimation (roll, pitch, and yaw or quaternion) is based on [Madgwick 2011].

In general, the relation between the calibrated measurement, $\mathbf{c}$, and un-calibrated measurement, $\mathbf{u}$, can be assumed as:

$$\mathbf{c} = \mathbf{K}\mathbf{u} - \mathbf{b} \tag{C.1}$$

where $\mathbf{K}$ is a matrix defining the sensor gain and $\mathbf{b}$ is the sensor bias.

The objective is to find $\mathbf{K}$ and $\mathbf{b}$ to avoid disturbances and measurement errors.

C.1.1 Accelerometer calibration

Accelerometer measures the acceleration forces and it is useful for determining the direction of Earth's gravity. We know the fact that the magnitude of Earth's gravity field is constant in everywhere and every direction. For a calibrated sensor, the measured magnitude will be constant for all orientations of the sensor. That is Equation C.2 will be hold and $\mathbf{K}$ and $\mathbf{b}$ can be found as the solution of Equation C.3.

$$m = \|\mathbf{K}\mathbf{u} - \mathbf{b}\| \tag{C.2}$$

Therefore, the principle of accelerometer calibration is to solve optimal problem as follow:

$$\min_{\mathbf{K},\mathbf{b}} \sum_{i} (m - \|\mathbf{K}\mathbf{u}_i - \mathbf{b}\|^2) \tag{C.3}$$

where m is the magnitude of Earth's gravity field, $\mathbf{u_i}$ represents the un-calibrated sensor measurement at the i'th orientation.

The calibration result of robot's IMU is shown in Figure C.1.

C.1.2 Gyroscope calibration

Gyroscope is a device to measure angular velocities. The principle of gyroscope calibration is to solve the following problem:

$$\min_{k_a} \left[r - T \sum_{t=0}^{n} (k_a u_{a,t} - b_{a,t}) \right]^2, \quad a = x, y, z \tag{C.4}$$

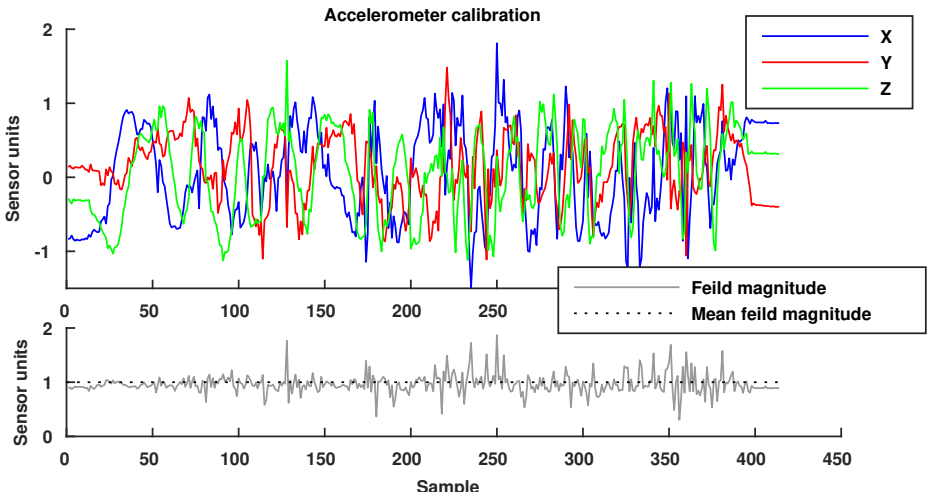


Figure C.1 – Accelerometer calibration

where $b_{a,t}$ is the bias at time t, k_a is sensor gain, T is the sampling period, r is the known angle of rotation, $u_{a,t}$ is the un-calibrated gyroscope measurement at time t.

The calibration results of gyroscope for X, Y, and Z axes are shown in Figure C.2, Figure C.3, and Figure C.4, respectively.

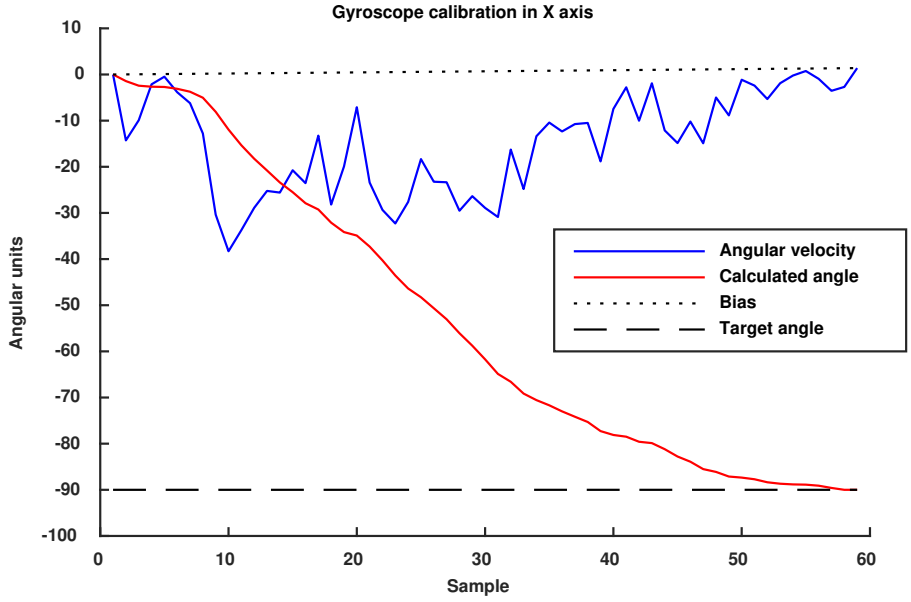


Figure C.2 – Gyroscope calibration in X axis

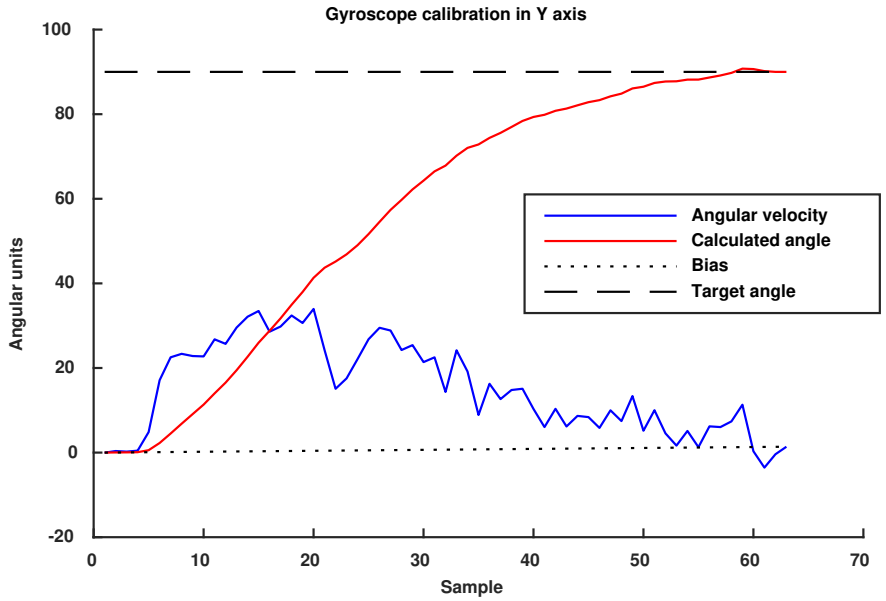


Figure C.3 – Gyroscope calibration in Y axis

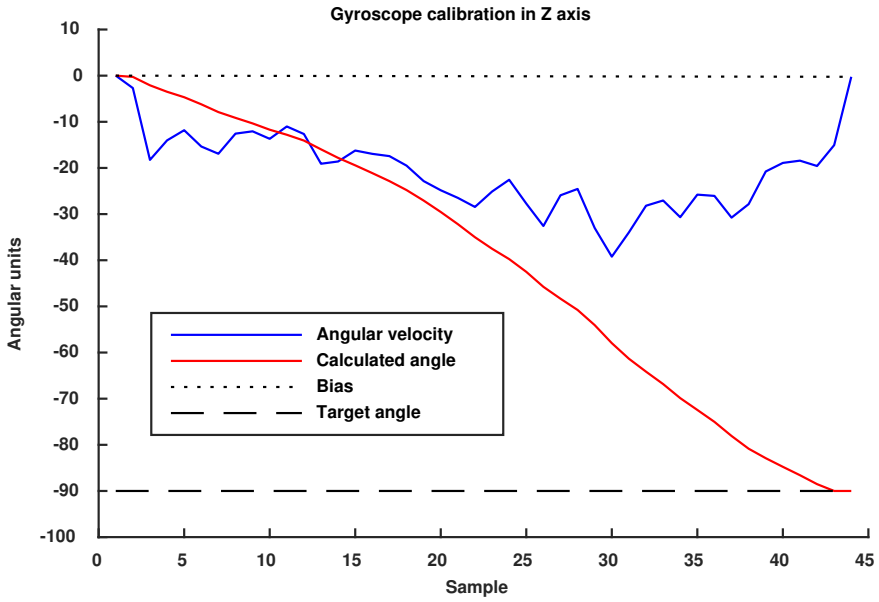


Figure C.4 – Gyroscope calibration in Z axis

C.1.3 Magnetometer calibration

Magnetometer is a device which measures magnetism-direction, strength of magnetic field and it is useful for determining the magnetic north direction. The calibration of magnetometer can be carried out by two methods. The first one is to use

the optimal problem as in accelerometer calibration and the second one is to use ellipse fit. Note that, magnetometer has to be rotated to collect good un-calibrated data. In this thesis, ellipse fit method has been used to calibrate magnetometer. First of all, un-calibrated data are stored by rotating magnetometer in all directions. The calibration results of magnetometer (ellipse fit method) are shown in Figure C.5 and Figure C.6. Following Figure C.5, original data (un-calibrated data) are red points. An ellipse is found by ellipsoid fit algorithm. Finally, compensated coefficients are interpolated and calibrated data are shown in Figure C.6.

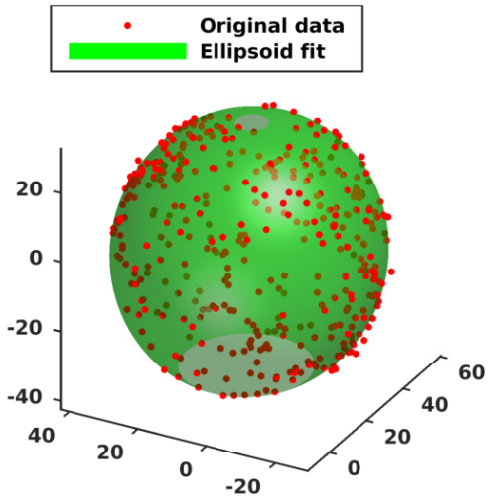


Figure C.5 – Magnetometer calibration - Ellipse fit

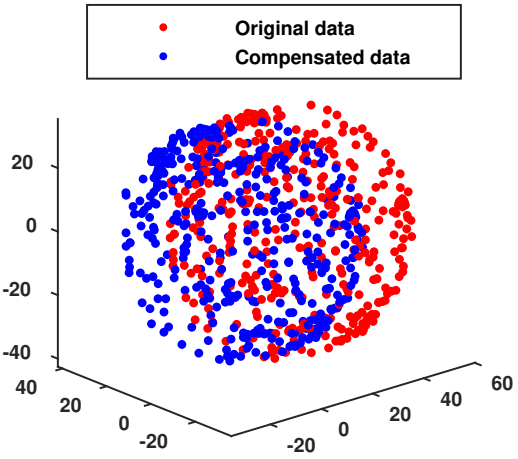


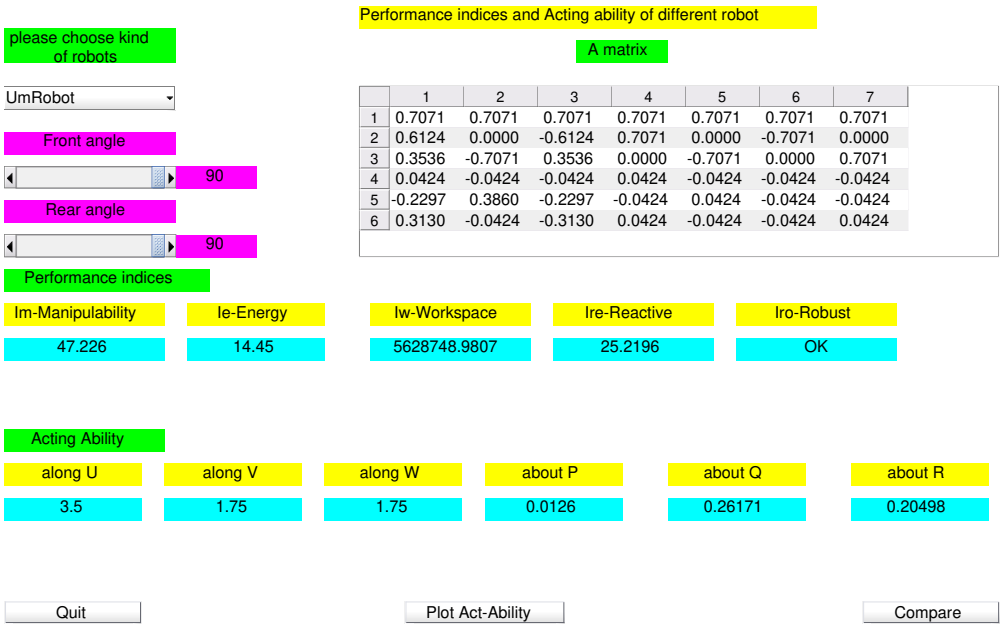
Figure C.6 – Magnetometer calibration

Appendix D

Appendix

D.1 Toolbox of configuration matrix evaluation

The toolbox is used to evaluate different configurations of different kinds of robots and was built in Matlab environment. The main page of the toolbox is displayed in Figure D.1. We can choose different kinds of robot (Ball robot, SamoS robot, Cube robot, Umbrella Robot) and also different configurations of the same robot (Umbrella Robot with different angles). For Umbrella robot, we can see clearly the directions of thrusters as Figure D.2. The performance indices and acting abilities are displayed on the main screen and configuration matrix, **A**, as well. We can plot acting abilities of a configuration in bar graph by *'Plot Act-Ability'* button. By clicking on *'Compare'* button, we go into comparison screen (Figure



	1	2	3	4	5	6	7
1	0.7071	0.7071	0.7071	0.7071	0.7071	0.7071	0.7071
2	0.6124	0.0000	-0.6124	0.7071	0.0000	-0.7071	0.0000
3	0.3536	-0.7071	0.3536	0.0000	-0.7071	0.0000	0.7071
4	0.0424	-0.0424	-0.0424	0.0424	-0.0424	-0.0424	-0.0424
5	-0.2297	0.3860	-0.2297	-0.0424	0.0424	-0.0424	-0.0424
6	0.3130	-0.0424	-0.3130	0.0424	-0.0424	-0.0424	0.0424

Figure D.1 – Toolbox for configuration evaluation: main page

D.3). Two configurations will be chosen to make a comparison of performance indices and acting abilities in bar graphs as in Figure D.4 for Cube robot.

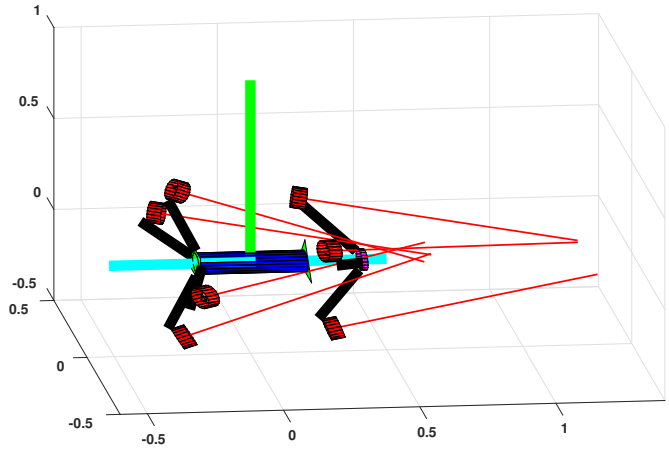


Figure D.2 – Umbrella robot model

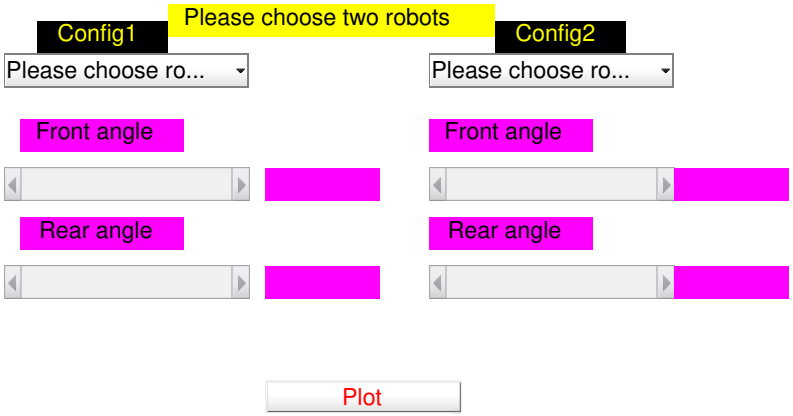


Figure D.3 – Toolbox for configuration evaluation: comparison page

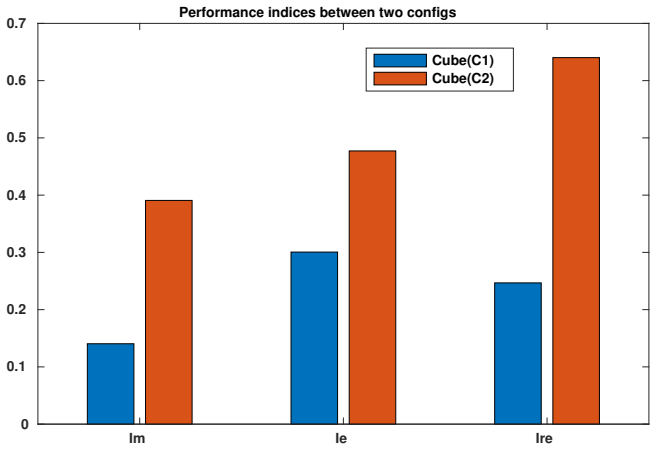


Figure D.4 – A comparison between $\mathbf{C}^1$ and $\mathbf{C}^2$ configurations

Bibliography

[Abbass 2001] Hussein A Abbass, Ruhul Sarker and Charles Newton. *PDE: a Pareto-frontier differential evolution approach for multi-objective optimization problems.* In Evolutionary Computation, 2001. Proceedings of the 2001 Congress on, volume 2, pages 971–978. IEEE, 2001. (Cited on page 52.)

[Adorno 2010] Bruno Vilhena Adorno, Philippe Fraisse and Sébastien Druon. *Dual position control strategies using the cooperative dual task-space framework.* In IROS'10: International Conference on Intelligent Robots and Systems, pages 3955–3960. IEEE, 2010. (Cited on page 57.)

[Agravante 2016] Don Joven Agravante, Alexander Sherikov, Pierre-Brice Wieber, Andrea Cherubini and Abderrahmane Kheddar. *Walking pattern generators designed for physical collaboration.* In Robotics and Automation (ICRA), 2016 IEEE International Conference on, pages 1573–1578. IEEE, 2016. (Cited on page 57.)

[Andersson 2019] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings and Moritz Diehl. *CasADi – A software framework for nonlinear optimization and optimal control.* Mathematical Programming Computation, vol. 11, no. 1, pages 1–36, 2019. (Cited on page 31.)

[Bertsekas 1995] Dimitri P. Bertsekas. Nonlinear programming. Athena Scientific, 1995. (Cited on page 38.)

[Bhat 2000] Sanjay P. Bhat and Dennis S. Bernstein. *A topological obstruction to continuous global stabilization of rotational motion and the unwinding phenomenon.* Systems & Control Letters, vol. 39, no. 1, pages 63 – 70, 2000. (Cited on page 27.)

[BlueRobotics] BlueRobotics. *BlueRobotics.* https://bluerobotics.com/. (Cited on pages 4 and 82.)

[Bodson 2002] Marc Bodson. *Evaluation of optimization methods for control allocation.* Journal of Guidance, Control, and Dynamics, vol. 25, no. 4, pages 703–711, 2002. (Cited on page 34.)

[Bordignon 1996] Kenneth A Bordignon. *Constrained control allocation for systems with redundant control effectors.* PhD thesis, Virginia Polytechnic Institute and State University Blacksburg, 1996. (Cited on page 34.)

[Boyd 2004] Stephen Boyd and Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004. (Cited on pages 37, 38 and 47.)

[Branke 2008] Jürgen Branke, Kalyanmoy Deb and Kaisa Miettinen. Multiobjective optimization: Interactive and evolutionary approaches, volume 5252. Springer Science & Business Media, 2008. (Cited on pages 7, 51 and 52.)

[Breivik 2006] Morten Breivik and Thor I Fossen. *A unified control concept for autonomous underwater vehicles.* In 2006 American Control Conference, pages 7–pp. IEEE, 2006. (Cited on page 122.)

[Brockett 1983] Roger W Brockett*et al.* *Asymptotic stability and feedback stabilization.* Differential geometric control theory, vol. 27, no. 1, pages 181–191, 1983. (Cited on page 28.)

[Caccia 2000] M Caccia and G Veruggio. *Guidance and control of a reconfigurable unmanned underwater vehicle.* Control engineering practice, vol. 8, no. 1, pages 21–37, 2000. (Cited on page 98.)

[Chankong 2008] Vira Chankong and Yacov Y Haimes. Multiobjective decision making: theory and methodology. Courier Dover Publications, 2008. (Cited on pages 45, 47, 48 and 49.)

[Chaturvedi 2011] N. A. Chaturvedi, A. K. Sanyal and N. H. McClamroch. *Rigid-Body Attitude Control.* IEEE Control Systems, vol. 31, no. 3, pages 30–51, June 2011. (Cited on page 27.)

[Chocron 2008] Olivier Chocron and Hervé Mangel. *Reconfigurable magnetic-coupling thrusters for agile AUVs.* In 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3172–3177. IEEE, 2008. (Cited on page 98.)

[Chocron 2013] Olivier Chocron, Urbain Prieur and Laurent Pino. *A validated feasibility prototype for AUV reconfigurable magnetic coupling thruster.* IEEE/ASME Transactions on Mechatronics, vol. 19, no. 2, pages 642–650, 2013. (Cited on page 98.)

[Chocron 2018] Olivier Chocron, Emanuel P Vega and Mohamed Benbouzid. *Dynamic reconfiguration of autonomous underwater vehicles propulsion system using genetic optimization.* Ocean Engineering, vol. 156, pages 564–579, 2018. (Cited on page 98.)

[Coello 2004] Carlos A Coello Coello, Gregorio Toscano Pulido and M Salazar Lechuga. *Handling multiple objectives with particle swarm optimization.* IEEE Transactions on evolutionary computation, vol. 8, no. 3, pages 256–279, 2004. (Cited on page 52.)

[Comex] Comex. *Comex.* https://comex.fr/. (Cited on page 15.)

[Dang 2019] Huu-Tho Dang, Lionel Lapierre, Rene Zapata, Pascal Lepinay and Benoit Ropars. *Configuration Matrix Design of Over-Actuated Marine Systems.* In OCEANS 2019-Marseille, pages 1–10. IEEE, 2019. (Cited on pages 3, 56, 113 and 123.)

[Daudelin 2018] Jonathan Daudelin, Gangyuan Jing, Tarik Tosun, Mark Yim, Hadas Kress-Gazit and Mark Campbell. *An integrated system for perception-driven autonomy with modular robots.* Science Robotics, vol. 3, no. 23, page eaat4983, 2018. (Cited on page 98.)

[De Novi 2009] G De Novi, Claudio Melchiorri, JC García, PJ Sanz, Pere Ridao and Gabriel Oliver. *A new approach for a reconfigurable autonomous underwater vehicle for intervention.* In 2009 3rd Annual IEEE Systems Conference, pages 23–26. IEEE, 2009. (Cited on page 98.)

[Deb 2002] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal and TAMT Meyarivan. *A fast and elitist multiobjective genetic algorithm: NSGA-II.* IEEE transactions on evolutionary computation, vol. 6, no. 2, pages 182–197, 2002. (Cited on page 52.)

[Dennis 2014] Louise A Dennis, Michael Fisher, Jonathan M Aitken, Sandor M Veres, Yang Gao, Affan Shaukat and Guy Burroughes. *Reconfigurable autonomy.* KI-Künstliche Intelligenz, vol. 28, no. 3, pages 199–207, 2014. (Cited on page 98.)

[Diebel 2006] James Diebel. *Representing attitude: Euler angles, unit quaternions, and rotation vectors.* Matrix, vol. 58, no. 15-16, pages 1–35, 2006. (Cited on page 147.)

[Durham 1993] Wayne C Durham. *Constrained control allocation.* Journal of Guidance, Control, and Dynamics, vol. 16, no. 4, pages 717–725, 1993. (Cited on page 34.)

[Elliott 2016] Christopher Michael Elliott. *A Stochastic Distributed Control Allocation Method Using Probability Collectives.* PhD thesis, 2016. (Cited on page 34.)

[Ende 2001] Barbara Anne Ende. *3D mapping of underwater caves.* IEEE Computer Graphics and Applications, vol. 21, no. 2, pages 14–20, 2001. (Cited on page 17.)

[Fairfield 2006] Nathaniel Fairfield, George Kantor and David Wettergreen. *Towards particle filter SLAM with three dimensional evidence grids in a flooded subterranean environment.* In Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., pages 3575–3580. IEEE, 2006. (Cited on page 16.)

[Fairfield 2007] Nathaniel Fairfield, George Kantor and David Wettergreen. *Real-time SLAM with octree evidence grids for exploration in underwater tunnels.* Journal of Field Robotics, vol. 24, no. 1-2, pages 03–21, 2007. (Cited on page 16.)

[Ferreau 2014] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock and Moritz Diehl. *qpOASES: A parametric active-set algorithm for quadratic programming.* Mathematical Programming Computation, vol. 6, no. 4, pages 327–363, 2014. (Cited on page 120.)

[Fonseca 1993] Carlos M Fonseca, Peter J Fleming*et al. Genetic Algorithms for Multiobjective Optimization: FormulationDiscussion and Generalization.* In Icga, volume 93, pages 416–423. Citeseer, 1993. (Cited on page 52.)

[FontaineDeVaucluse] FontaineDeVaucluse. *Fontaine De Vaucluse.* http://www.plongeesout.com/sites/provence/vaucluse/vaucluse (Cited on pages 3, 13 and 14.)

[Ford 2007a] Derek Ford. *Jovan Cvijić and the founding of karst geomorphology.* Environmental Geology, vol. 51, no. 5, pages 675–684, 2007. (Cited on page 10.)

[Ford 2007b] Derek Ford and Paul D Williams. Karst hydrogeology and geomorphology. John Wiley & Sons, 2007. (Cited on page 10.)

[Fossen 2006] T. I. Fossen and T. A. Johansen. *A Survey of Control Allocation Methods for Ships and Underwater Vehicles*. In 2006 14th Mediterranean Conference on Control and Automation, pages 1–6, June 2006. (Cited on page 34.)

[Fossen 2009] Thor I Fossen, Tor Arne Johansen and Tristan Perez. A survey of control allocation methods for underwater vehicles. INTECH Open Access Publisher, 2009. (Cited on page 34.)

[Fossen 2011] Thor I Fossen. Handbook of marine craft hydrodynamics and motion control. John Wiley & Sons, 2011. (Cited on pages 28, 57, 169 and 170.)

[Fukuda 1988a] Toshio Fukuda and Seiya Nakagawa. *Approach to the dynamically reconfigurable robotic system*. Journal of Intelligent and Robotic Systems, vol. 1, no. 1, pages 55–72, 1988. (Cited on page 97.)

[Fukuda 1988b] Toshio Fukuda and Seiya Nakagawa. *Dynamically reconfigurable robotic system*. In Proceedings. 1988 IEEE International Conference on Robotics and Automation, pages 1581–1586. IEEE, 1988. (Cited on page 97.)

[Gembicki 1975] F Gembicki and Y Haimes. *Approach to performance and sensitivity multiobjective optimization: The goal attainment method*. IEEE Transactions on Automatic control, vol. 20, no. 6, pages 769–771, 1975. (Cited on pages 50 and 73.)

[Gleick 1993] Peter H Gleick. *Water in crisis*. Pacific Institute for Studies in Dev., Environment & Security. Stockholm Env. Institute, Oxford Univ. Press. 473p, vol. 9, 1993. (Cited on pages 3 and 10.)

[Gourmelen 2018] Guillaume Gourmelen. *Conception de un robot sous marin a geometrie variable*, 2018. (Cited on page 99.)

[Grechi 2016] Simone Grechi and Andrea Caiti. *Comparison between Optimal Control Allocation with Mixed Quadratic & Linear Programming Techniques*. IFAC-PapersOnLine, vol. 49, no. 23, pages 147 – 152, 2016. 10th IFAC Conference on Control Applications in Marine SystemsCAMS 2016. (Cited on page 58.)

[Härkegård 2003] Ola Härkegård. *Backstepping and control allocation with applications to flight control*. PhD thesis, Linköpings universitet, 2003. (Cited on pages 34 and 115.)

[Holland 1975] John H Holland. *Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence*. Ann Arbor, MI: University of Michigan Press, pages 439–444, 1975. (Cited on page 51.)

[Horn 1994] Jeffrey Horn, Nicholas Nafpliotis and David E Goldberg. *A niched Pareto genetic algorithm for multiobjective optimization*. In Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence.,

Proceedings of the First IEEE Conference on, pages 82–87. Ieee, 1994. (Cited on page 52.)

[Hu 2007] Yonghui Hu, Long Wang, Wei Zhao, Qi Wang and Le Zhang. *Modular design and motion control of reconfigurable robotic fish*. In 2007 46th IEEE Conference on Decision and Control, pages 5156–5161. IEEE, 2007. (Cited on page 98.)

[Huet 2016] Cécile Huet and Franco Mastroddi. *Autonomy for underwater robots - A European perspective*. Autonomous Robots, vol. 40, no. 7, pages 1113–1118, 2016. (Cited on page 11.)

[Isidori 1989] Alberto Isidori. Nonlinear control systems design. Elsevier, 1989. (Cited on page 30.)

[Johansen 2013] Tor A. Johansen and Thor I. Fossen. *Control allocation,ÄîA survey*. Automatica, vol. 49, no. 5, pages 1087 – 1103, 2013. (Cited on pages 34, 35, 57, 58 and 114.)

[Jourde 2007] Hervé Jourde, Axel Roesch, Vincent Guinot and Vincent Bailly-Comte. *Dynamics and contribution of karst groundwater to surface flow during Mediterranean flood*. Environmental Geology, vol. 51, no. 5, pages 725–730, 2007. (Cited on page 13.)

[Kereluk 2017] Jason A Kereluk and M Reza Emami. *Task-based optimization of reconfigurable robot manipulators*. Advanced Robotics, vol. 31, no. 16, pages 836–850, 2017. (Cited on page 97.)

[Khalil 2002] Hassan Khalil. Nonlinear systems, volume 3. Prentice hall Upper Saddle River, NJ, 2002. (Cited on pages 29 and 30.)

[Kharrat 2015] Houssem Kharrat. Optimization of thruster configuration for swimming robots. Master's thesis, Rice University, 2015. (Cited on page 58.)

[Knowles 2000] Joshua D Knowles and David W Corne. *Approximating the nondominated front using the Pareto archived evolution strategy*. Evolutionary computation, vol. 8, no. 2, pages 149–172, 2000. (Cited on page 52.)

[Krstic 1995] Miroslav Krstic, Petar V Kokotovic and Ioannis Kanellakopoulos. Nonlinear and adaptive control design. John Wiley & Sons, Inc., 1995. (Cited on pages 30 and 31.)

[Kuipers 1999] Jack B Kuipers. Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality. Princeton university press, 1999. (Cited on pages 26, 27 and 32.)

[Kumar 1981] A Kumar and KJ Waldron. *The workspaces of a mechanical manipulator*. Journal of Mechanical Design, vol. 103, no. 3, pages 665–672, 1981. (Cited on page 58.)

[Lapierre 2006a] Lapierre. *Underwater robots part I: Current systems and problem pose*. In Mobile Robots: towards New Applications. IntechOpen, 2006. (Cited on page 11.)

[Lapierre 2006b] Lapierre. *Underwater robots part II: Existing solutions and open issues*. In Mobile Robots: towards New Applications. InTechOpen, 2006. (Cited on page 21.)

[Lapierre 2006c] Lionel Lapierre, D Soetanto and Antonio Pascoal. *Nonsingular path following control of a unicycle in the presence of parametric modelling uncertainties*. International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal, vol. 16, no. 10, pages 485–503, 2006. (Cited on page 167.)

[Lapierre 2007] Lionel Lapierre and Didik Soetanto. *Nonlinear path-following control of an AUV*. Ocean engineering, vol. 34, no. 11-12, pages 1734–1744, 2007. (Cited on page 28.)

[Lapierre 2008] Lionel Lapierre and Bruno Jouvencel. *Robust nonlinear path-following control of an AUV*. IEEE Journal of Oceanic Engineering, vol. 33, no. 2, pages 89–102, 2008. (Cited on pages 31 and 56.)

[Lapierre 2009] Lionel Lapierre. *Robust diving control of an AUV*. Ocean Engineering, vol. 36, no. 1, pages 92–104, 2009. (Cited on page 56.)

[Lapierre 2016] Lionel Lapierre. Aleyin: An underneath robotic journey - robotic systems for karst exploration. 2016. (Cited on pages 3 and 15.)

[Lasbouygues 2017] Adrien Lasbouygues, Silvain Louis, Benoît Ropars, Luc Rossi, Herve Jourde, Hélène Délas, Pierre Balordi, Rémi Bouchard, Mehdi Dighouth, Marc Dugrenot*et al. Robotic mapping of a karst aquifer*. 2017. (Cited on page 17.)

[Levine 2010] William S Levine. The control systems handbook: Control system applications. CRC press, 2010. (Cited on page 57.)

[Liu 2016] Jinguo Liu, Xin Zhang and Guangbo Hao. *Survey on research and development of reconfigurable modular robots*. Advances in Mechanical Engineering, vol. 8, no. 8, page 1687814016659597, 2016. (Cited on page 97.)

[Lopes 2017] Luís Lopes, Norbert Zajzon, Balázs Bodo, Stephen Henley, Gorazd Žibret and Tatjana Dizdarevic. *UNEXMIN: developing an autonomous underwater explorer for flooded mines*. Energy Procedia, vol. 125, pages 41–49, 2017. (Cited on page 16.)

[Louis 2017] Silvain Louis, Lionel Lapierre, Karen Godary-Dejean, Yadpiroon Onmek, Thomas Claverie and Sebastien Villéger. *Quaternion based control for robotic observation of marine diversity*. In OCEANS, 2017. (Cited on page 133.)

[Low 2007] KH Low and Junzhi Yu. *Development of modular and reconfigurable biomimetic robotic fish with undulating fin*. In 2007 IEEE International Conference on Robotics and Biomimetics (ROBIO), pages 274–279. IEEE, 2007. (Cited on page 98.)

[Madgwick 2011] Sebastian OH Madgwick, Andrew JL Harrison and Ravi Vaidyanathan. *Estimation of IMU and MARG orientation using a gradient descent algorithm*. In 2011 IEEE international conference on rehabilitation robotics, pages 1–7. IEEE, 2011. (Cited on page 175.)

[Martins-Encarnação 2002] P Martins-Encarnação. *Nonlinear path following control system for ocean vehicles*. PhD thesis, Tese (Doutorado) Universidade Técnica de Lisboa, 2002. (Cited on page 163.)

[Meister 2013] Eugen Meister, Alexander Gutenkunst and Paul Levi. *Dynamics and control of modular and self-reconfigurable robotic systems*. Int. J. Adv. Intell. Syst, vol. 6, no. 1, 2013. (Cited on page 98.)

[Miettinen 1999] K. Miettinen. Nonlinear multiobjective optimization. Kluwer Academic Publishers, 1999. (Cited on pages 45, 47, 48, 49 and 50.)

[Mintchev 2012] Stefano Mintchev, Cesare Stefanini, Alexis Girin, Stefano Marrazza, Stefano Orofino, Vincent Lebastard, Luigi Manfredi, Paolo Dario and Frederic Boyer. *An underwater reconfigurable robot with bioinspired electric sense*. In 2012 IEEE International Conference on Robotics and Automation, pages 1149–1154. IEEE, 2012. (Cited on page 98.)

[Mirjalili 2016] Seyedali Mirjalili, Shahrzad Saremi, Seyed Mohammad Mirjalili and Leandro dos S Coelho. *Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization*. Expert Systems with Applications, vol. 47, pages 106–119, 2016. (Cited on page 52.)

[Moreno 2014] Eduardo Moreno and Shu-Yun Chung. *SeaDrone: A modular and reconfigurable underwater robot for task optimization*. In OCEANS 2014-TAIPEI, pages 1–7. IEEE, 2014. (Cited on page 98.)

[Morten Breivik 2009] Thor I. Fossen Morten Breivik. Underwater vehicles. InTechOpen, 2009. (Cited on page 160.)

[Murata 2007] Satoshi Murata and Haruhsa Kurokawa. *Self-reconfigurable robots*. IEEE Robotics & Automation Magazine, vol. 14, no. 1, pages 71–78, 2007. (Cited on page 98.)

[Nakamura 1987] Yoshihiko Nakamura, Hideo Hanafusa and Tsuneo Yoshikawa. *Task-priority based redundancy control of robot manipulators*. The International Journal of Robotics Research, vol. 6, no. 2, pages 3–15, 1987. (Cited on page 57.)

[Nocedal 2006] Jorge Nocedal and Stephen Wright. Numerical optimization. Springer Science & Business Media, 2006. (Cited on pages 37, 38, 39, 42, 44 and 121.)

[Odetti 2017] Angelo Odetti, Marco Bibuli, Giorgio Bruzzone, Massimo Caccia, Edoardo Spirandelli and Gabriele Bruzzone. *e-URoPe: a reconfgurable AUV/ROV for man-robot underwater cooperation*. IFAC-PapersOnLine, vol. 50, no. 1, pages 11203–11208, 2017. (Cited on page 99.)

[Paden 1988] Brad Paden and Shankar Sastry. *Optimal kinematic design of 6R manipulators*. The International Journal of Robotics Research, vol. 7, no. 2, pages 43–61, 1988. (Cited on pages 58 and 63.)

[Page 2000] Anthony B Page and Marc L Steinberg. *A closed-loop comparison of control allocation methods*. In Proc. of AIAA Guidance, Navigation, and Control Conference, pages 1760–1770, 2000. (Cited on page 34.)

[Park 1994] Frank C Park and Roger W Brockett. *Kinematic dexterity of robotic mechanisms*. The International Journal of Robotics Research, vol. 13, no. 1, pages 1–15, 1994. (Cited on page 58.)

[Pham 2018] Tu-Hoa Pham, Stéphane Caron and Abderrahmane Kheddar. *Multi-contact Interaction Force Sensing From Whole-Body Motion Capture*. IEEE Transactions on Industrial Informatics, vol. 14, no. 6, pages 2343–2352, 2018. (Cited on page 57.)

[Pierrot 1998] F. Pierrot, M. Benoit and P. Dauchez. *Optimal thruster configuration for omni-directional underwater vehicles. SamoS: a Pythagorean solution*. In OCEANS '98 Conference Proceedings, volume 2, pages 655–659 vol.2, Sep 1998. (Cited on pages 58, 61, 76 and 112.)

[Prabakaran 2018] Veerajagadheswar Prabakaran, Mohan Rajesh Elara, Thejus Pathmakumar and Shunsuke Nansai. *Floor cleaning robot with reconfigurable mechanism*. Automation in Construction, vol. 91, pages 155–165, 2018. (Cited on page 98.)

[Prats 2012] Mario Prats, David Ribas, Narcís Palomeras, Juan Carlos García, Volker Nannen, Stephan Wirth, José Javier Fernández, Joan P Beltrán, Ricard Campos, Pere Ridao*et al.* *Reconfigurable AUV for intervention missions: a case study on underwater object recovery*. Intelligent Service Robotics, vol. 5, no. 1, pages 19–31, 2012. (Cited on page 98.)

[Pugi 2018] Luca Pugi, Benedetto Allotta and Marco Pagliai. *Redundant and reconfigurable propulsion systems to improve motion capability of underwater vehicles*. Ocean Engineering, vol. 148, pages 376–385, 2018. (Cited on page 98.)

[Rawlings 2017] James Blake Rawlings, David Q Mayne and Moritz Diehl. Model predictive control: theory, computation, and design, volume 2. Nob Hill Publishing Madison, WI, 2017. (Cited on page 31.)

[Ribas 2011] David Ribas, Pere Ridao, Lluís Magí, Narcís Palomeras and Marc Carreras. *The Girona 500, a multipurpose autonomous underwater vehicle*. In Oceans 2011 IEEE-Spain, pages 1–5. IEEE, 2011. (Cited on page 98.)

[Richmond 2018] Kristof Richmond, Chris Flesher, Laura Lindzey, Neal Tanner and William C Stone. *SUNFISH®: A human-portable exploration AUV for complex 3D environments*. In OCEANS 2018 MTS/IEEE Charleston, pages 1–9. IEEE, 2018. (Cited on page 17.)

[Ropars 2015] Benoit Ropars, Adrien Lasbouygues, Lionel Lapierre and David Andreu. *Thruster's dead-zones compensation for the actuation system of an underwater vehicle*. In Control Conference (ECC), 2015 European, pages 741–746. IEEE, 2015. (Cited on pages 12, 35 and 58.)

[Ropars 2018] B. Ropars, L. Lapierre, A. Lasbouygues, D. Andreu and R. Zapata. *Redundant actuation system of an underwater vehicle*. Ocean Engineering, vol. 151, pages 276 – 289, 2018. (Cited on pages 57, 75 and 113.)

[Ruzika 2005] Stefan Ruzika and Margaret M Wiecek. *Approximation methods in multiobjective programming.* Journal of optimization theory and applications, vol. 126, no. 3, pages 473–501, 2005. (Cited on page 50.)

[Sahl 2010] Jason W Sahl, Nathaniel Fairfield, J Kirk Harris, David Wettergreen, William C Stone and John R Spear. *Novel microbial diversity retrieved by autonomous robotic exploration of the world's deepest vertical phreatic sinkhole.* Astrobiology, vol. 10, no. 2, pages 201–213, 2010. (Cited on page 16.)

[Sanz 2010] Pedro J Sanz, Mario Prats, Pere Ridao, David Ribas, Gabriel Oliver and Alberto Ortiz. *Recent progress in the RAUVI project: A reconfigurable autonomous underwater vehicle for intervention.* In Proceedings ELMAR-2010, pages 471–474. IEEE, 2010. (Cited on page 98.)

[Schaffer 1985] J David Schaffer. *Multiple objective optimization with vector evaluated genetic algorithms.* In Proceedings of the First International Conference on Genetic Algorithms and Their Applications, 1985. Lawrence Erlbaum Associates. Inc., Publishers, 1985. (Cited on page 52.)

[Schmitz 1988] Donald Schmitz. *The CMU reconfigurable modular manipulator system.* 1988. (Cited on page 97.)

[Skulstad 2018] Robert Skulstad, Guoyuan Li, Houxiang Zhang and Thor I Fossen. *A Neural Network Approach to Control Allocation of Ships for Dynamic Positioning.* IFAC-PapersOnLine, vol. 51, no. 29, pages 128–133, 2018. (Cited on page 34.)

[Srinivas 1994] Nidamarthi Srinivas and Kalyanmoy Deb. *Muiltiobjective optimization using nondominated sorting in genetic algorithms.* Evolutionary computation, vol. 2, no. 3, pages 221–248, 1994. (Cited on page 52.)

[Stephan 2017] J. Stephan and W. Fichter. *Fast Exact Redistributed Pseudoinverse Method for Linear Actuation Systems.* IEEE Transactions on Control Systems Technology, vol. PP, no. 99, pages 1–8, 2017. (Cited on page 58.)

[Stone 2007] WC Stone. *Design and deployment of a 3D autonomous subterranean submarine exploration vehicle.* In Proceedings UUST07, Conference on Unmanned, Un-tethered Submersable Technology, Durham, NH, 2007. (Cited on page 16.)

[Stoy 2010] Kasper Stoy, David Brandt and David Johan Christensen. Self-reconfigurable robots: an introduction. Mit Press, 2010. (Cited on page 98.)

[SubseaTech] SubseaTech. *Tortuga Robot.* (Cited on page 113.)

[Taylor 1972] Taylor and Mann. Advanced calculus. John Wiley & Sons, 1972. (Cited on page 163.)

[Taylor 2008] Charles J Taylor and Earl A Greene. *Hydrogeologic characterization and methods used in the investigation of karst hydrology.* Field techniques for estimating water fluxes between surface water and ground water, edited by: Rosenberry, DO and LaBaugh, JW, US Geological Survey, Reston, Virginia (EUA), pages 71–114, 2008. (Cited on pages 3 and 13.)

[Thakker 2014] Rohan Thakker, Ajinkya Kamat, Sachin Bharambe, Shital Chiddarwar and Kishor M Bhurchandi. *Rebis-reconfigurable bipedal snake robot.* In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 309–314. IEEE, 2014. (Cited on page 98.)

[Transeth 2008] Aksel Andreas Transeth, Remco I Leine, Christoph Glocker, Kristin Ytterstad Pettersen and Pål Liljebäck. *Snake robot obstacle-aided locomotion: Modeling, simulations, and experiments.* IEEE Transactions on Robotics, vol. 24, no. 1, pages 88–104, 2008. (Cited on page 113.)

[Trefethen 1997] Lloyd N Trefethen and David Bau III. Numerical linear algebra, volume 50. Siam, 1997. (Cited on page 72.)

[Vega 2016] Emanuel Pablo Vega. *Task-based design and optimization of reconfigurable propulsion systems for autonomous underwater vehicles.* PhD thesis, Université de Bretagne occidentale-Brest, 2016. (Cited on page 98.)

[Xiang 2015] Xianbo Xiang, Lionel Lapierre and Bruno Jouvencel. *Smooth transition of AUV motion control: From fully-actuated to under-actuated configuration.* Robotics and Autonomous Systems, vol. 67, pages 14–22, 2015. (Cited on page 122.)

[Yim 2007] Mark Yim, Wei-Min Shen, Behnam Salemi, Daniela Rus, Mark Moll, Hod Lipson, Eric Klavins and Gregory S Chirikjian. *Modular self-reconfigurable robot systems [grand challenges of robotics].* IEEE Robotics & Automation Magazine, vol. 14, no. 1, pages 43–52, 2007. (Cited on page 97.)

[Yoshikawa 1985a] T. Yoshikawa. *Dynamic manipulability of robot manipulators.* In Proceedings. 1985 IEEE International Conference on Robotics and Automation, volume 2, pages 1033–1038, Mar 1985. (Cited on page 57.)

[Yoshikawa 1985b] Tsuneo Yoshikawa. *Manipulability of robotic mechanisms.* The international journal of Robotics Research, vol. 4, no. 2, pages 3–9, 1985. (Cited on page 61.)

[Yoshikawa 1990] Tsuneo Yoshikawa. Foundations of robotics: analysis and control. Mit Press, 1990. (Cited on page 58.)

[Yuh 2000] Junku Yuh. *Design and control of autonomous underwater robots: A survey.* Autonomous Robots, vol. 8, no. 1, pages 7–24, 2000. (Cited on pages 11 and 113.)

[Zereik 2018] Enrica Zereik, Marco Bibuli, Nikola Mišković, Pere Ridao and António Pascoal. *Challenges and future trends in marine robotics.* Annual Reviews in Control, 2018. (Cited on pages 11 and 113.)

[Zhang 2007] Qingfu Zhang and Hui Li. *MOEA/D: A multiobjective evolutionary algorithm based on decomposition.* IEEE Transactions on evolutionary computation, vol. 11, no. 6, pages 712–731, 2007. (Cited on page 52.)

[Zitzler 1999] Eckart Zitzler. *Evolutionary algorithms for multiobjective optimization: Methods and applications.* 1999. (Cited on page 52.)

[Zitzler 2001] Eckart Zitzler, Marco Laumanns and Lothar Thiele. *SPEA2: Improving the strength Pareto evolutionary algorithm.* TIK-report, vol. 103, 2001. (Cited on page 52.)