Control of an Underwater Swimming Manipulator, With Compensation for Reaction Forces and Hydrostatic Forces

Morten Fyhn Amundsen

Master of Science in Cybernetics and Robotics

Submission date:	June 2017
Supervisor:	Kristin Ytterstad Pettersen, ITK
Co-supervisor:	Eleni Kelasidi, ITK
	Jørgen Sverdrup-Thygeson, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem description

The project is concerned with a class of robots called underwater swimming manipulators, or USMs. These are hybrid systems that merge features of underwater snake robots (USRs), robotic manipulator arms, and traditional thruster-equipped underwater robots, such as ROVs and AUVs. A USM has an appearance similar to that of a floating manipulator arm, and is equipped with thrusters in addition to movable joints. It can move freely in water either by undulatory locomotion, or by combining the undulation with thruster propulsion. USMs are designed to be highly maneuverable, while also having the ability to perform intervention tasks.

Most USMs—including the specific model covered here—have a number of joints that make them redundant with respect to the tasks that they perform. The hyperredundancy creates a very interesting control problem, providing both more flexibility and thus more challenges when it comes to motion planning and control. USM control must also take into account that the USM floats freely in the water—it is not attached to its environment. Control of the hyperreduntant, free-floating USM is the topic of this project.

1. Do a literature survey on modeling and control of redundant robot manipulators, especially for underwater robots.

2. Develop a control system for USM manipulation tasks.

3. Implement the control system in MATLAB/Simulink.

4. Validate the control system through simulations with Vortex Studio.

The report shall be written in English and edited as a research report including Abstract, Introduction with motivation, literature survey, contributions of the project work, and the outline of the report. This is followed by the chapters describing the results of the project work, simulation results and corresponding discussion, and a conclusion including a proposal for further work.

Assignment given:	January 9, 2017
Deadline:	June 5, 2017
Supervisor:	Kristin Y. Pettersen
Co-supervisors:	Jørgen Sverdrup-Thygeson
	Eleni Kelasidi

Abstract

The *underwater swimming manipulator* (USM) is a new class of robots that combines the *underwater snake robot*'s biologically inspired shape with the thruster actuators typical of other underwater robots. This combination makes USMs highly flexible and facilitates for inspection, maintenance, and repair in otherwise inaccessible environments. However, because the USM floats freely in the water, any motion of the manipulator will cause a reaction that affects the position and orientation of the base, and hydrostatic forces will cause rotations of the entire vehicle. This thesis proposes a USM control scheme that accounts for reaction forces and hydrostatic forces. The approach is based on a method originally developed for robots in space, using the *generalized Jacobian matrix*. The generalized Jacobian takes base-manipulator coupling into account and allows manipulation without actuating the base coordinates. Applying this to a USM differs from previous USM control methods, which actively control the base link to allow fixed-base control of the end-effector. The previous methods require many thrusters, and the power to drive them. This thesis investigates if USM control with the generalized Jacobian is feasible, despite the differences between underwater and space robots. The complete control system is implemented in *MATLAB/Simulink* and verified in the *Vortex Studio* real-time hydrodynamic simulator. Without hydrostatic forces, the system is found to be useful even without sensor feedback, with small tracking errors. With nonzero hydrostatic forces, sensor feedback is required for acceptable accuracy, and position feedback was found to give significantly better precision than velocity feedback. The thesis demonstrates that the generalized Jacobian can be used for efficient USM control, and is applicable to any underwater vehicle manipulator system, including underwater snake robots without thrusters. For USMs, this means that longer missions on battery power are possible and that the thrusters are freed to other tasks.

Sammendrag

Svømmende undervannsmanipulatorer (USM-er) er en ny klasse roboter som kombinerer den biologisk inspirerte formen til *undervanns slangeroboter* med thrustere som er vanlige for andre undervannsroboter. Denne kombinasjonen gjør USM-er svært fleksible og legger til rette for inspeksjon, vedlikehold og reparasjon i ellers utilgjengelige omgivelser. Ettersom USM-er flyter fritt i vannet vil enhver leddbevegelse skape reaksjonskrefter som påvirker posisjonen og rotasjonen til basen, og hydrostatiske krefter vil få hele farkosten til å rotere. Denne avhandlingen foreslår et styresystem for USM-er som tar høyde for reaksjonskrefter og hydrostatiske krefter. Tilnærmingen er basert på en metode utviklet for roboter i verdensrommet, som benytter den *generaliserte jacobimatrisen*. Den generaliserte jacobimatrisen tar hensyn til koplingskrefter mellom base og manipulator-arm, og muliggjør manipulering uten å aktuere basens koordinater. Å anvende dette på en USM skiller seg fra tidligere styringsmetoder for USM-er, som aktivt regulerer basen for å tillate fast base-styring av armen. De tidligere metodene krever mange thrustere, og effekt til å drive dem. Denne avhandlingen undersøker om USM-styring med den generaliserte jacobimatrisen er mulig, tross forskjellene mellom roboter under vann og i verdensrommet. Det fullstendige styresystemet er implementert i *MATLAB/Simulink* og verifisert i *Vortex Studio*, en sanntids hydrodynamikk-simulator. Uten hydrostatiske krefter er styresystemet nyttig selv uten tilbakekopling, med små reguleringsavvik. Med hydrostatiske krefter er tilbakekopling nødvendig for å oppnå akseptabel nøyaktighet, og posisjonstilbakekopling har vist seg å være vesentlig mer presist enn hastighetstilbakekopling. Avhandlingen viser at den generaliserte jacobimatrisen kan gi effektiv USM-styring, og er anvendelig på ethvert undervanns farkost-manipulator-system, inkludert slangeroboter uten thrustere. For USM-er betyr dette at lengre oppdrag blir mulige, og thrusterne blir fri til å utføre andre oppgaver.

Contents

List of Tables

List of Figures

List of Acronyms

AUV	autonomous underwater vehicle
CAD	computer-aided design
CLIK	closed-loop inverse kinematics
COB	center of buoyancy
COM	center of mass
DOF	degree of freedom
DVL	Doppler velocity log
GJM	generalized Jacobian matrix
GNSS	global navigation satellite system
IMU	inertial measurement unit
INS	inertial navigation system
LBL	long baseline
PD	proportional-derivative
PID	proportional-integral-derivative
PIW	path-independent workspace
RMS	root mean square

RNS	reaction null-space
ROV	remotely operated vehicle
SLAM	simultaneous localization and mapping
SNAME	The Society of Naval Architects and Marine Engineers
SVD	singular value decomposition
USBL	ultra-short baseline
USM	underwater swimming manipulator
USR	underwater snake robot
UVMS	underwater vehicle-manipulator system
VM	virtual manipulator

List of Symbols

n	Number of joints
n_d	Number of controllable DOFs
n_t	Number of thrusters
$\mathbb{R}$	The set of real numbers
$\mathcal{S}(3) \subset \mathbb{R}^3$	A sphere
$\mathcal{H} \subset \mathbb{R}^4$	The set of unit quaternions
$SO(3) \subset \mathbb{R}^{3\times 3}$	The special orthogonal group
$\mathcal{F}_I$	The inertial frame
$\mathcal{F}_0$	The USM base frame
$\mathcal{F}_i$	The USM joint i frame
$\mathcal{F}_n$	The USM end-effector link frame
$\mathcal{F}_E$	The USM end-effector tool frame
$\{{}^a\hat{\boldsymbol{x}}_b, {}^a\hat{\boldsymbol{y}}_b, {}^a\hat{\boldsymbol{z}}_b\}$	Normalized Cartesian basis for $\mathcal{F}_b$ expressed in $\mathcal{F}_a$
${}^a\boldsymbol{p}_b \in \mathbb{R}^3$	Position of $\mathcal{F}_b$ given in $\mathcal{F}_a$
${}^a\boldsymbol{R}_b \in SO(3)$	Rotation from $\mathcal{F}_a$ to $\mathcal{F}_b$
${}^a\boldsymbol{T}_b \in \mathbb{R}^{4\times 4}$	Homogeneous transformation from $\mathcal{F}_a$ to $\mathcal{F}_a$
${}^a\boldsymbol{\eta}_{b,c}$	Pose of $\mathcal{F}_c$ relative to $\mathcal{F}_b$, expressed in $\mathcal{F}_a$ [1]

[1]The target set depends on the representation of orientation. Using Euler angles, it is $\mathbb{R}^3 \times \mathcal{S}(3)$, and using unit quaternions, it is $\mathbb{R}^3 \times \mathcal{H}$.

${}^{a}\boldsymbol{v}_{b} = ({}^{a}u_{b}, {}^{a}v_{b}, {}^{a}w_{b})^{\mathrm{T}} \in \mathbb{R}^{3}$	Linear velocities of $\mathcal{F}_b$ wrt. $\mathcal{F}_a$
${}^{a}\boldsymbol{\omega}_{b} = ({}^{a}p_{b}, {}^{a}q_{b}, {}^{a}r_{b})^{\mathrm{T}} \in \mathbb{R}^{3}$	Angular velocities of $\mathcal{F}_b$ wrt. $\mathcal{F}_a$
${}^{a}\boldsymbol{\nu}_{b} = ({}^{a}\boldsymbol{v}_{b}^{\mathrm{T}}, {}^{a}\boldsymbol{\omega}_{b}^{\mathrm{T}})^{\mathrm{T}} \in \mathbb{R}^{6}$	Velocity twist (linear and angular) of $\mathcal{F}_b$ wrt. $\mathcal{F}_a$
${}^{a}\boldsymbol{\tau} = ({}^{a}\boldsymbol{f}^{\mathrm{T}}, {}^{a}\boldsymbol{m}^{\mathrm{T}})^{\mathrm{T}} \in \mathbb{R}^{6}$	Wrench (forces and moments) expressed in $\mathcal{F}_a$
${}^{a}\boldsymbol{f}_{\mathrm{grav}} \in \mathbb{R}^{3}$	Force of gravity expressed in $\mathcal{F}_a$
${}^{a}\boldsymbol{f}_{\mathrm{buoy}} \in \mathbb{R}^{3}$	Force of buoyancy expressed in $\mathcal{F}_a$
${}^{a}\boldsymbol{g} \in \mathbb{R}^{6}$	Hydrostatic restoring forces expressed in $\mathcal{F}_a$
$\boldsymbol{q} = (q_1, \ldots, q_n) \in \mathbb{R}^{n}$	Manipulator joint positions
$\dot{\boldsymbol{q}} \in \mathbb{R}^{n}$	Manipulator joint velocities
$\boldsymbol{t} = (t_1, \ldots, t_m) \in \mathbb{R}^{m}$	Task space positions
$\dot{\boldsymbol{t}} \in \mathbb{R}^{m}$	Task space velocities
$\boldsymbol{h} = (\eta, \boldsymbol{\epsilon}^{\mathrm{T}})^{\mathrm{T}}$	Quaternion base orientation
$\boldsymbol{J}$	Arbitrary Jacobian matrix
$\boldsymbol{J}^{\dagger}$	Pseudoinverse of the Jacobion
$\boldsymbol{J}_{g}^{S}(\boldsymbol{q}) \in \mathbb{R}^{6\times n}$	Spatial geometric manipulator Jacobian
$\boldsymbol{J}_{g}^{0}(\boldsymbol{q}) \in \mathbb{R}^{6\times n}$	Base geometric manipulator Jacobian
$\boldsymbol{J}_{g}^{E}(\boldsymbol{q}) \in \mathbb{R}^{6\times n}$	End-effector geometric manipulator Jacobian
$\boldsymbol{J}^{*}(\boldsymbol{q}, {}^{I}\boldsymbol{R}_{0}) \in \mathbb{R}^{6\times n}$	Generalized Jacobian matrix
$\boldsymbol{S}(\boldsymbol{p}) \in \mathbb{R}^{3\times 3}$	Skew-symmetric matrix of vector $\boldsymbol{p} \in \mathbb{R}^{3}$
$\boldsymbol{Ad}\left({}^{a}\boldsymbol{T}_{b}\right) \in \mathbb{R}^{6\times 6}$	Adjoint map of the transformation ${}^{a}\boldsymbol{T}_{b}$
$\boldsymbol{X}_{i}'$	Base joint twist
$\boldsymbol{X}_{i}^{i}$	Body joint twist
$\boldsymbol{B} = \left(\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{n_t}\right) \in \mathbb{R}^{n_d \times n_t}$	Thrust configuration matrix
$\boldsymbol{u} \in \mathbb{R}^{n_t}$	Thruster force vector
m_i	Mass of USM link i
m_{tot}	Total USM mass
W	Total USM weight

∇_i	Displacement of USM link i
∇_{tot}	Total USM displacement
B	Total USM buoyancy
g	Acceleration of gravity
ρ	Density of water
$\mathbf{I}$	The identity matrix
$\boldsymbol{K}_{\text{clik}} \in \mathbb{R}^{m\times m}$	CLIK gain
$\boldsymbol{K}_p \in \mathbb{R}^{3\times 3}$	Orientation controller proportional gain
$\boldsymbol{K}_d \in \mathbb{R}^{3\times 3}$	Orientation controller derivative gain
$\lambda_d, \lambda_w, \lambda_n$	Inverse kinematics scaling factors
$\boldsymbol{\sigma} = (\sigma_{\min}, \dots, \sigma_{\max})^{\text{T}}$	Singular values of the Jacobian
$H(\boldsymbol{q})$	Performance criterion for joint limit avoidance
$\boldsymbol{W} \in \mathbb{R}^{n\times n}$	Joint velocity weight matrix

Chapter 1

Introduction

This introductory chapter will briefly provide context for the class of robots discussed in this thesis and give motivation and a description of the problem to be solved. A literature review summarizes existing relevant knowledge, and establishes the foundation of the later control system. The scope of the work is then defined through a list of assumptions. Finally, the contributions of the thesis are defined and elaborated.

1.1 Motivation

Underwater robots exist in numerous forms and have a multitude of applications. Applications include seafloor mapping and geological sampling in research and science; construction, inspection, maintenance and repair of subsea installations in the oil and gas industry; and search and disposal of mines for the military [1]. Figure 1.1 shows examples of different classes of underwater robots.

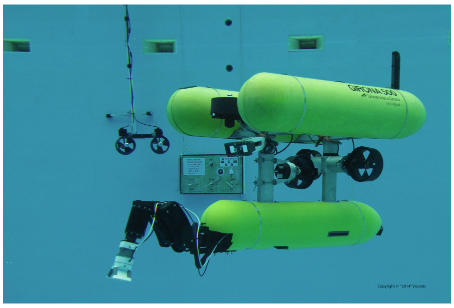
Some tasks, especially complex intervention tasks that have yet to be automated, and also some observation tasks, are suited for remotely operated vehicles (ROVs) (Figures 1.1a and 1.1b), which are tethered to the surface and usually teleoperated [2]. Other tasks, such as wide area surveying and mapping, are usually done by autonomous underwater vehicles (AUVs) (Figures 1.1c and 1.1d), as they can operate for extended periods of time without human involvement [1].

The underwater swimming manipulator (USM) is a new class of underwater robots

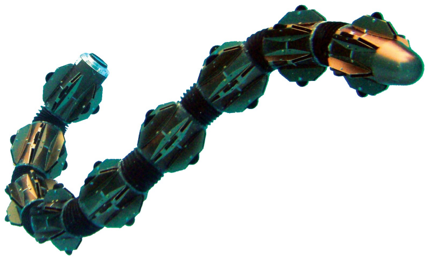
(a) AC-CESS AC-ROV 100 miniature ROV

(b) Sperre SUB-fighter 10k ROV

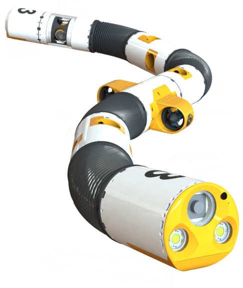
(c) Kongsberg REMUS 100 AUV



(d) Girona 500 AUV



(e) HiBot ACM-R5H USR



(f) Eelume USM

Figure 1.1: Examples of classes of underwater robots.

that combines a bio-inspired snake-like appearance with thruster actuators [3]. USMs are currently only in development by *Eelume*, and represent a novel field of robotics research. The Eelume USM is shown in Figure 1.1f. USMs are a development of the underwater snake robot (USR), which is a snake-like underwater robot designed to "swim" like an eel or a sea snake [4]. Figure 1.1e gives an example of a USR.

USMs are intended to combine the range of AUVs, the accessibility of small ROVs, and the intervention capabilities of work class ROVs. The shape of a USM gives it access in confined and cluttered environments, and it can adapt to different tasks by assuming an appropriate configuration [3]. For long-distance transportation, the USM assumes a fully extended shape to reduce drag, while for intervention it can use its motorized joints to move like an industrial manipulator arm. Another prospect is to use USMs as "resident" robots: Robots that live on site underwater, with a charging and docking station to return to between missions.

This thesis will investigate how to achieve accurate and flexible USM manipulation with reduced power consumption compared to today's methods. Previous USM and underwater vehicle-manipulator system (UVMS) control has used thrusters and joint motors for manipulation, which is energy-inefficient due to the high power consumption of the thrusters. In contrast to the existing methods, this thesis will seek a solution that does not rely on thruster usage, ultimately saving power and extending the possible mission duration.

1.2 Problem description

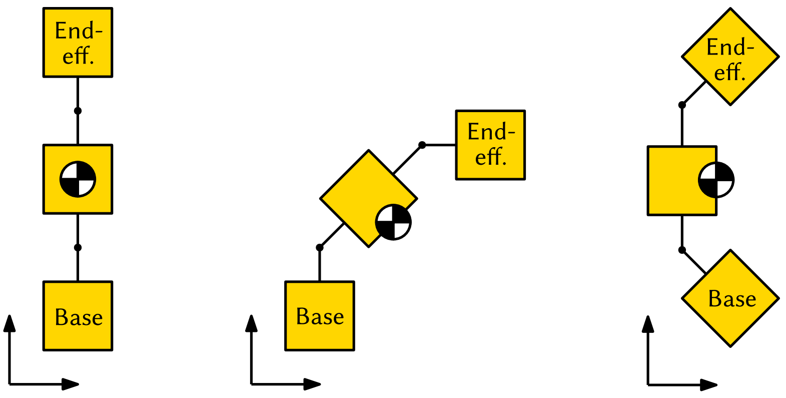
This thesis will develop a manipulation control system for the underwater swimming manipulator class of robots. A USM floats freely in the water, in contrast to an industrial robotic manipulator which is fixed to the ground. Because of this, the USM responds differently to three types of forces in particular:

1. **Reaction forces:** Moving the joints of the USM will induce reaction forces that disturb the position and orientation of the manipulator base [5]. The disturbance occurs because the base is floating freely in the water. By contrast, the base of an industrial manipulator is firmly fixed to its environment and is not disturbed by joint motion. Figure 1.2 compares the effects of joint motion and reaction forces on fixed-base and floating-base manipulators.

2. **Hydrostatic forces:** Gravity “pulls” the center of mass (COM) down, and buoyancy “pushes” the center of buoyancy (COB) up. For a USM, its total COM and COB are not necessarily aligned, and this causes rotational hydrostatic restoring forces on the USM [6, Ch. 4]. Because the COM and COB locations depend on the joint configuration of the USM, the resulting restoring forces also depend on the joint configuration.

3. **Hydrodynamic forces:** The manipulator moves through water, and is subject to hydrodynamic effects such as drag forces and increased inertia from added mass [7, Ch. 10].

The effects of the forces listed above are present for all underwater manipulators, but not to the same extent. Robots that have a large “body” with a smaller manipulator attached to it will not be strongly affected, because of the large inertia of the base. Intervention-ROVs and -AUVs are typical examples, such as those in Figures 1.1b and 1.1d.



(a) Initial configuration (b) Fixed base (c) Floating base

Figure 1.2: Differences between fixed-base and floating-base manipulation. (a) shows the initial configuration, and (b) and (c) shows final configurations after identical joint motion. In (b), the base is fixed, and this causes the total COM, denoted ⊕, to move. In (c), the base is floating, and reaction forces on the base cause it to move, while the location of the COM remains the same.

USRs and USMs are more strongly affected than traditional UVMSs. The snake robots have a base link that is only a small fraction of the total vehicle’s inertia and size.

As a result, both reaction forces and hydrostatic forces that occur while moving the manipulator will affect them much more than ROVs and AUVs. It is therefore vital to compensate for these effects when attempting manipulation tasks.

The title of the thesis describes its core aim: "Control of an Underwater Swimming Manipulator, With Compensation for Reaction Forces and Hydrostatic Forces". The necessary elements for the realization of such a control system will be developed, including a suitable model, the control scheme itself, and concerns necessary for practical implementation. The proposed solutions apply to many classes of robots, but will be discussed in the context of USMs here.

The control scheme in this thesis will not be designed to counteract hydrodynamic effects. The thesis will, however, investigate the significance of hydrodynamics in comparison to hydrostatics and reaction forces, and to what extent the hydrodynamic forces subside at the low velocities typical for precision manipulation. The quadratic terms in hydrodynamic added mass and damping [7, Ch. 10] suggest that low velocities lead to small hydrodynamic forces.

1.3 Literature review

This section reviews central literature in areas relevant to the problem description above, including modeling, inverse kinematics, and free-floating manipulation.

A thorough overview of how to derive forward kinematics for UVMSs, through a series of homogeneous transformation matrices, is given in [7, Ch. 4–5]. A USM is—kinematically speaking—simply a UVMS with a tiny base, and the same methods are directly applicable to USMs. An example of applying UVMS modeling methods to USMs is given in [8].

A kinematic and dynamic model for underwater snake robots is derived in [9]. It is general in the sense that it does not assume constant link lengths or masses, and it also models added thrusters. This makes it a generalization of the USR model of [10], in which link lengths and masses are assumed constant, and there are no added effectors. The model of [10] is, in turn, a generalization of the land-based snake robot model given in [11].

The control system proposed in this thesis is based on kinematics combined with the mass and inertia properties of the robot. It is therefore not necessary to use a full dynamic model—it is sufficient to have a kinematic model and knowledge of basic vehicle parameters.

1.3.1 Inverse kinematic control and redundancy resolution

Inverse kinematic control is the problem of determining the joint positions or velocities (or even accelerations) needed to fulfill a given task, such as reaching a given end-effector pose. For redundant manipulators, there will be an infinite number of solutions for a task, and a method for choosing one of them is needed. This is referred to as *redundancy resolution* and is illustrated with an example in Figure 1.3. A survey of some widely adopted methods is given in [12].

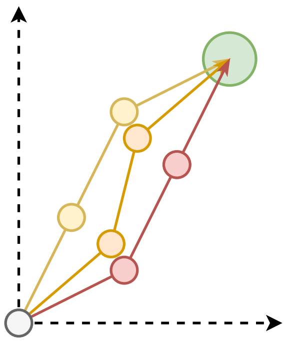


Figure 1.3: Effects of kinematic redundancy: A 3 joint manipulator performing a 2 DOF end-effector positioning task. The gray dot at the origin represents the first joint, the arrow heads represent end-effectors, and the green circle is the desired end-effector position. Yellow, orange, and red are used to display different joint configurations that all accomplish the same task—this is possible due to the kinematic redundancy.

The most common inverse kinematic redundancy resolution methods are based on the Jacobian matrix of the manipulator. Two simple methods are the Jacobian transpose [13] and the more widely used Jacobian pseudoinverse [14]. Siciliano [12] explains that the Jacobian transpose may be useful when the computational load is of particular importance and that it avoids the numerical instability that Jacobian inverse methods exhibit at kinematic singularities. The Jacobian pseudoinverse is also easy to implement and computationally cheap, and is optimal in a minimum least-square joint velocity sense [12]. This optimality condition is useful for the control system investigated in this thesis, but the lack of singularity avoidance is problematic, due to the large and spurious velocities singularities can cause.

Another way to resolve redundancy is by defining more than one task. This not only resolves but exploits the mechanical redundancy. A common approach is *task priority redundancy resolution*, which is based on filtering a secondary task through the null-space of the primary task. The filtering gives a strict priority between the tasks—the secondary task is only fulfilled to the extent possible without interfering with the primary task. The method is described in [15] in the context of general robot manipulators, and [16] in the context of UVMSs. Task priority can also be used for redundancy resolution alone, by defining an arbitrary secondary task, but this may lead to algorithmic singularities [12].

Singularity-robust task-priority has been applied to a USM in [3], where it is used to coordinate the motion of the base and end-effector, by actuating both the joints and the thrusters. The primary task was six degree of freedom (DOF) control of the end-effector, and the secondary task was stationkeeping of the base. This is an example of applying traditional fixed-base inverse kinematics to a floating base robot, and it takes stationkeeping errors into account.

An *augmented* Jacobian can also resolve redundancy. The idea is to add a task by altering the Jacobian directly, in such a way that it becomes square and invertible [12]. It then becomes possible to use a regular matrix inverse as opposed to the pseudoinverse, but the strict priority between tasks disappears. The augmented Jacobian also restricts the number of possible subtasks considerably.

1.3.2 Singularity avoidance

Several singularity robust inverse kinematic methods exist. An alternative to the Jacobian pseudoinverse, termed the singularity-robust inverse, is presented in [17], which gives feasible joint velocity solutions at and near singular points. Another singularity-avoidance technique, *damped least-squares*, based on damping the pseudoinverse Jacobian near singular points, is suggested in [18]. *Numerical filtering* is an improvement on this, presented in [19], which maintains the singularity robustness but reduces tracking error near singularities by only damping the relevant joint velocities. The latter two are very relevant—the first for its straightforward implementation and the second for its improved tracking near singular points.

Furthermore, [20] studies existing singularity-robust methods applied to redundant manipulators and proposes a new method to overcome the problem of algorithmic singularities. This method is, however, only applicable to task-priority-based inverse kinematics.

Chirikjian and Burdick [21] present a solution that avoids the Jacobian altogether: The macroscopic configuration of the manipulator is here defined by a *backbone curve*, which is a curve in space that represents the curvature of the manipulator itself. The inverse kinematics simplify to determining how the curve changes over time. A fitting procedure calculates joint actuator references that make the manipulator assume the shape of the curve. The method applies to many classes of robot manipulators, including inextensible rigid-link snake robots such as the USM.

1.3.3 Robot manipulators in space

There has been much research done on the behavior and control of vehicles with manipulators in outer space. Because both vehicle and manipulator float freely in vacuum, without significant external forces, the system is more complex than a fixed-base manipulator on earth.

In [22], the kinematics of a satellite-mounted manipulator are examined. When the manipulator's joints move, the satellite base experiences translational and rotational reaction forces. These effects are derived for a manipulator with rigid links, both for a free-floating satellite and an attitude-controlled satellite. The authors mention the possibility of using these terms as a feed-forward signal to a controller, and present a reaction moment compensation command that can cancel the attitude disturbance caused by the manipulator. The feed-forward requires knowledge of the joint positions along with their first and second derivatives. This can be hard to measure or estimate accurately, but assuming sufficiently small deviations from the planned path, one may use the planned joint trajectory and its derivatives instead [22].

The generalized Jacobian matrix (GJM) is introduced and discussed in [23] and [5], amongst others. The GJM is, as the name implies, a generalization of the Jacobian matrix, and can be used for inverse kinematic manipulator control on a floating base, as it takes momentum conservation and base-manipulator coupling into account. [23] derives the GJM, explains some practical considerations, and suggests directions for future study. It is made clear that the GJM can be used for inverse kinematics just like a conventional Jacobian matrix: direct inversion if it is square and non-singular; pseudoinversion otherwise. An in-orbit experiment has been conducted to validate the approach in a realistic environment, which successfully verified both GJM and reaction null-space (RNS)-based control methods [24].

An alternative approach is presented in [25]. It is based on the concept of a virtual manipulator (VM)—an imagined, shorter manipulator that represents the actual reachable workspace of the real manipulator when accounting for the coupling between the base and the manipulator. The base of the virtual manipulator is fixed to the COM of the real vehicle-manipulator-system, and its end-effector is usually, but not necessarily, chosen to coincide with the actual end-effector. The COM does not move, assuming no external forces, and the VM can be controlled as a fixed-base manipulator. Any given joint configuration will result in the same position of the end-effector of the VM and the real manipulator. However, this method only considers conservation of linear momentum [26], and is more suitable for systems with active attitude control. The USM can perform attitude control with its thrusters, but the GJM does not have this requirement and is, therefore, a better alternative.

Many subsequent publications have used the GJM, the VM approach, and other methods to solve floating-base manipulation problems. Some consider only free-floating bases; some assume active attitude control; and some design for platforms with active attitude and position control. Examples are [27]–[31].

A decoupling of manipulator and base dynamics is achieved in [32] by the use of the RNS. The method differs from those based on the GJM or a VM in that it attempts to carry out manipulation in a way that does not produce reaction forces. This eliminates the need for any attitude or position correction, but the downside is that working only inside the reaction null-space may restrict the workspace too much.

The size of the workspace along with a measure of manipulability for floating manipulators is defined and discussed in [33]. As the reachable workspace for a floating manipulator is smaller than that of a fixed-base manipulator, knowing its bounds is especially interesting. Several types of reachable workspaces based on different assumptions are defined. For the USM, knowledge of the reachable workspace can be used to determine which tasks can be carried out without a repositioning of the robot as a whole. Another way to determine workspace is given [25], based on VMs.

1.3.4 Hydrodynamic effects on underwater manipulators

Although modeling and prediction of hydrodynamic effects is outside the scope of this thesis, it is still something to keep in mind for future work. A hydrodynamic model for a cylindrical, single-link underwater manipulator is developed in [34]. The authors identified and modeled the presence of state-dependent hydrodynamic coefficients. In comparison to traditional constant-coefficient models, the new model gave a "significant improvement in modeling accuracy" [35, p. 463]. The model was used in a model-based coordinated vehicle/manipulator control scheme in [34]. Using the OTTER vehicle for experiments, they compared the cases of

- no vehicle control,
- pure feedback control,
- pure feed-forward control, and
- combined feedback and feed-forward control.

with the conclusion that "Experimental results showed that substantial performance improvements could be realized in the control of an underwater arm/vehicle system by incorporating model-based feed-forward information about the hydrodynamic coupling into the control of the system." [34, p. 1215]. For a USM used for pure manipulation, without actuating the thrusters, this accurate cylinder-based model is applicable. If the thrusters are actuated, however, the hydrodynamic behavior must be expected to change, possibly invalidating the model. Further analysis of the hydrodynamic properties of underwater manipulators with thrusters is required, but outside the scope of this thesis.

1.4 Assumptions

This section defines assumptions that will be adopted throughout the thesis.

Assumption 1. The USM links are rigid bodies.

Assumption 2. The USM link masses and displaced volumes are constant.

Assumption 3. No water currents or wave forces act on the USM.

Assumption 4. The initial velocities of the USM base and joints are zero.

Assumption 5. The thrust configuration matrix is well-conditioned in 6 degrees of freedom.

Assumption 6. The USM as a whole is neutrally buoyant.

Assumption 7. The COM and COB of each link is known.

Assumption 8. All joints on the USM are 1 DOF revolute joints.

Remark 1. Assumption 4 can be fulfilled by giving the USM a few seconds to stabilize in the water.

Remark 2. Assumption 6 implies that the USM does not float or sink. It may, however, experience rotational hydrostatic forces if the COM and COB are out of alignment.

Remark 3. Assumption 8 is not very restrictive, as multi-DOF joints can be modeled as successive single-DOF joints.

1.5 Contributions

The following is a list of the main contributions of this thesis:

- The core contribution is applying GJM-based control to underwater robotic manipulators. GJM control is previously only used in space robotics, yet this thesis demonstrates its potential for other, non-space applications, including USMs. Its primary advantage is the ability to control the manipulator without regard to the base coordinates, and therefore without needing to actuate the base, and it has been proven to provide highly accurate control. This also makes it possible to perform manipulation with thruster-less USRs, which have not previously been used for manipulation tasks underwater. (Section 3.1.)

- A general kinematic model for robotic manipulators is applied to the USM and verified to be valid without the need for simplification or any other adaptation, thus proving that standard modeling frameworks also apply to USMs. (Chapter 2.)

- A simple velocity controller and an adaptation of an existing position controller have been formulated for use in conjunction with GJM control. The controllers are implemented and simulated as a study of how hydrostatic disturbances and unmodeled hydrodynamic effects can be counteracted. (Sections 3.2.5 and 3.2.6.)

- An attitude controller has been adapted from a known stationkeeping algorithm. The controller is used in conjunction with predictions of the vehicle orientation at hydrostatic equilibrium to stabilize the vehicle quickly, with minimal thruster usage. With this solution, the thrusters of the USM can be utilized. The thruster damping functionality is modular and possible to deactivate. (Sections 3.2.3 and 3.2.7.)

- Existing joint limit avoidance methods have been improved so that they provide identical functionality independent of how large the range of the joints are. (Section 3.4.)

- A thrust allocation algorithm has been adapted to the USM. Thrust allocation of USMs is different from conventional underwater robots, as the configuration of the thrusters changes when the joints move. (Section 3.5.)

- All parts of the kinematic model and control system are implemented in MATLAB/Simulink. The core functionality of the implementation has been unit tested to assure its validity. (Appendix B.)

- A robust custom interface between MATLAB/Simulink and the Vortex Studio simulation software has been developed. The interface vectorizes inputs and outputs and guarantees that the simulator output is valid at all times. Output velocities and positions are transformed appropriately to allow outputs for arbitrary base and end-effector frames. (Section 4.2.)

- The control system has been verified through extensive simulations, comparing six variants of the control system on two different simulation models. (Chapter 4.)

- The main contribution of the thesis—GJM control of underwater vehicles—has been condensed into a research paper draft, to be submitted to the European Control Conference of 2018. (Appendix C.)

1.6 Thesis outline

After this introductory chapter, the outline of the thesis is as follows: Chapter 2 presents a kinematic model of the USM. Chapter 3 develops a control system for free-floating manipulation, based on the kinematic model. Chapter 4 runs extensive simulations of several variants of the control system, and discusses the results case-by-case. Chapter 5 draws conclusions, and suggests directions for future work.

Chapter 2

Kinematic model

This chapter defines the forward and inverse kinematics of the USM. A kinematic model is a description of the movement of a system without regard to underlying forces and inertias. A dynamic model, on the other hand, also includes the forces that act on the system and the mass-dependent accelerations they cause. The model presented here is based on [7] and [36], with some changes to notation. The model is general and applies to any snake robot or robotic manipulator with revolute joints. It can also easily be extended to accommodate prismatic joints. The generality of the model makes it applicable to any variation of floating robotic manipulators, and not exclusively to USMs. It can, however, be applied to USMs without modification.

2.1 Forward kinematics

This section will present the forward kinematic model of the USM: The spatial relationships between all manipulator links and joints as functions of the joint positions. All relevant reference frames and the transformations between them will be defined.

2.1.1 Reference frames

The links and joints of the USM are numbered as in [36]: Joints from 1 to n, and links from 0 to n. Joint i connects links $i - 1$ and i. The index of the base link is 0, and the index of the end-effector is n. Reference frames are fixed to the joints so that the joint's rotational axis aligns with one of the frame axes. Frame $\mathcal{F}_i$ has its origin on the axis of joint i and is fixed to link i. There is no joint 0. Hence, the base frame $\mathcal{F}_0$ is set to some convenient location fixed to the base link, such as its COM or COB. The end-effector has a separate frame $\mathcal{F}_E$, fixed to the end-effector link and therefore to frame $\mathcal{F}_n$. It is common to define the end-effector frame to be between the jaws of a gripper or something of similar effect. The relative locations of the reference frames are illustrated in Figure 2.1, and summarized in Table 2.1.

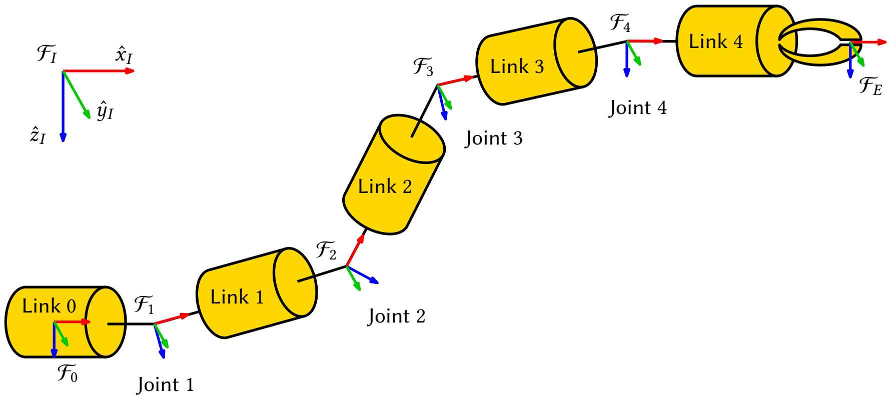


Figure 2.1: Notation for links, joints, and reference frames, shown on a 5-link manipulator. Note that $\mathcal{F}_4 = \mathcal{F}_n$.

The home position is defined so that all joint positions are zero, $\boldsymbol{q}_0 = (0, \ldots, 0)^{\mathrm{T}} \in \mathbb{R}^{n\times 1}$. When the USM is in its home position, all its coordinate frames are parallel in their x, y, and z axes. Their x axes are directed longitudinally (toward the end-effector), their y axes laterally (to the right when watching from behind and above), and their z axes normally (downwards when the robot is level). Figure 2.2 shows a joint frame in its home position and in an arbitrary position.

Finally, the inertial frame $\mathcal{F}_I$ is fixed to an arbitrary location in the environment, and its positive z axis points down.

2.1.2 Transformations between frames

Transformations between frames are expressed as homogeneous transformation matrices [37], as they allow concise notation and ease of programmatic implementation [36,

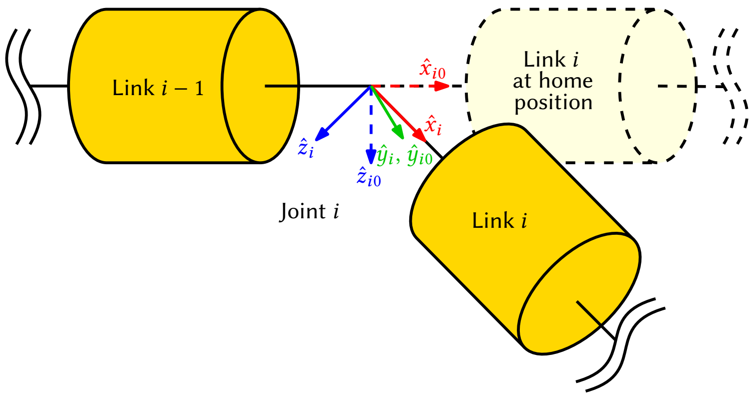


Figure 2.2: Close-up on a joint frame and its corresponding home-configuration frame.

Table 2.1: Summary of reference frames.

Frame	Description
$\mathcal{F}_I$	inertial frame (z-axis down)
$\mathcal{F}_0$	base link local frame
$\mathcal{F}_i$	joint i frame for $i \in \{0, 1, \ldots, n\}$
$\mathcal{F}_n$	end-effector joint frame
$\mathcal{F}_E$	end-effector tool frame

p. 17]. The homogeneous transformation from $\mathcal{F}_i$ to $\mathcal{F}_j$ is denoted ${}^i\boldsymbol{T}_j$. (In other words, ${}^i\boldsymbol{T}_j$ expresses the position and orientation of $\mathcal{F}_j$ relative to $\mathcal{F}_i$.) A transformation ${}^i\boldsymbol{T}_j$ is composed of a rotation matrix ${}^i\boldsymbol{R}_j \in \mathbb{R}^{3\times 3}$ and a translation ${}^i\boldsymbol{p}_j \in \mathbb{R}^3$. The three are related by

$$ {}^i\boldsymbol{T}_j = \begin{pmatrix} {}^i\boldsymbol{R}_j & {}^i\boldsymbol{p}_j \\ \boldsymbol{0} & 1 \end{pmatrix} \in \mathbb{R}^{4\times 4}. \tag{2.1} $$

The transformation ${}^{i-1}\boldsymbol{T}_i$ between successive joint frames depends only on the angle q_i of the ith joint and can be written

$$ {}^{i-1}\boldsymbol{T}_i(q_i) = \begin{pmatrix} {}^{i-1}\boldsymbol{R}_i(q_i) & {}^{i-1}\boldsymbol{p}_i \\ \boldsymbol{0} & 1 \end{pmatrix}, \tag{2.2} $$

where ${}^{i-1}\boldsymbol{p}_i$ is the position of $\mathcal{F}_i$ expressed in $\mathcal{F}_{i-1}$. The transformation ${}^0\boldsymbol{T}_E(\boldsymbol{q})$ from the base frame $\mathcal{F}_0$ to the end-effector frame $\mathcal{F}_E$ is a product of successive transformation matrices

$$ {}^0\boldsymbol{T}_E(\boldsymbol{q}) = {}^0\boldsymbol{T}_1(q_1)\,{}^1\boldsymbol{T}_2(q_2)\cdots{}^{n-1}\boldsymbol{T}_n(q_n)\,{}^n\boldsymbol{T}_E. \tag{2.3} $$

In the following, the notation is sometimes simplified by omitting the argument of the transformations, so that $\boldsymbol{T} \triangleq \boldsymbol{T}(\boldsymbol{q})$.

2.2 Differential kinematics

The forward kinematics define a map from the joint positions to e.g. the pose of the end-effector, usually relative to the base link as

$$ {}^0\boldsymbol{\eta}_E = \boldsymbol{k}(\boldsymbol{q}), \tag{2.4} $$

where the definition of the pose vector $\boldsymbol{\eta}$ depends on the chosen representation of orientation.

The tasks a USM carries out are defined in a task space—the space spanned out by the task variables. For the task of 6 DOF end-effector control, there are 6 task variables: Translation along three axes and rotation around three axes. However, the USM inputs are in joint space, not task space. The forward kinematics $\boldsymbol{k}(\boldsymbol{q})$ maps between these two spaces, with $\boldsymbol{k}(\boldsymbol{q})$ depending on the chosen task variables. Finding the necessary joint motion for a desired task space action requires its inverse $\boldsymbol{k}^{-1}(\boldsymbol{q})$, but $\boldsymbol{k}(\boldsymbol{q})$ is not generally invertible. Two options exist to solve the inverse kinematics:

1. Solve the inverse kinematics numerically, such as with the algorithm of [38] or the *IKFast* software module from OpenRAVE [39]. Such methods have the benefit of solving for joint positions directly but are time-consuming and complex in implementation, especially for elaborate or unusual robots, which makes it less desirable for a real-time system.

2. Solve the inverse kinematics at the differential level. The map from joint space velocities to task space velocities takes the form of a Jacobian matrix, the inverse of which is easier to compute (or approximate) than the position level forward kinematics. The drawback is that task space and joint space motions must be expressed as velocities, which adds complexity to systems controlled at the position level [40].

The remainder of this chapter presents variations of the manipulator Jacobian matrix.

2.2.1 The manipulator Jacobian matrix

The Jacobian matrix for a robot manipulator provides a map from its joint velocities to the linear and angular velocities of the end-effector.[1] There are multiple ways to express the velocities of the end-effector, and this affects the corresponding Jacobian matrix. Two useful choices are

- ${}^{0}\boldsymbol{\nu}_{0,E}$: the end-effector velocity relative to the base frame, expressed in the base frame, and
- ${}^{E}\boldsymbol{\nu}_{0,E}$: the end-effector velocity relative to the base frame, expressed in the end-effector frame.

The map from joint velocities to ${}^{0}\boldsymbol{\nu}_{0,E}$ is expressed with the base geometric Jacobian $\boldsymbol{J}_g^0$,

$$ {}^{0}\boldsymbol{\nu}_{0,E} = \boldsymbol{J}_g^0(\boldsymbol{q})\,\dot{\boldsymbol{q}}, \tag{2.5} $$

[1]It is possible to express the Jacobian for the velocities of any link, but the end-effector is usually the link of interest.

while the map from joint velocities to ${}^{E}\boldsymbol{\nu}_{0,E}$ is expressed with the end-effector geometric Jacobian[2] $\boldsymbol{J}_g^E$,

$$ {}^{E}\boldsymbol{\nu}_{0,E} = \boldsymbol{J}_g^E(\boldsymbol{q})\,\dot{\boldsymbol{q}}. \tag{2.6}$$

Following from its definition, the geometric Jacobian of a manipulator can be calculated by first deriving its forward kinematics vector $\boldsymbol{k}(\boldsymbol{q})$, and then analytically calculating partial derivatives of each element with respect to each joint variable. Explicitly calculating the partial derivatives is very laborious for manipulators with many joints. A method that scales better is given in [7] and repeated here. First, the *spatial* geometric Jacobian is written

$$\boldsymbol{J}_g^S = \begin{pmatrix} \boldsymbol{X}_1' & \boldsymbol{X}_2' & \dots & \boldsymbol{X}_n' \end{pmatrix} \tag{2.7}$$

where

$$\boldsymbol{X}_i' = \boldsymbol{Ad}\left({}^{0}\boldsymbol{T}_i\right)\,\boldsymbol{X}_i^i, \tag{2.8}$$

are the base joint twists,

$$\boldsymbol{Ad}\left({}^{i}\boldsymbol{T}_j\right) = \begin{pmatrix} {}^{i}\boldsymbol{R}_j & \boldsymbol{S}({}^{i}\boldsymbol{p}_j)\,{}^{i}\boldsymbol{R}_j \\ \boldsymbol{0} & {}^{i}\boldsymbol{R}_j \end{pmatrix}, \tag{2.9}$$

are the adjoint maps from the base to each joint,

$$\boldsymbol{X}_i^i = \begin{cases} (0,0,0,1,0,0)^{\mathrm{T}}, & \text{for } x\text{-axis rotation,} \\ (0,0,0,0,1,0)^{\mathrm{T}}, & \text{for } y\text{-axis rotation,} \\ (0,0,0,0,0,1)^{\mathrm{T}}, & \text{for } z\text{-axis rotation,} \end{cases} \tag{2.10}$$

are the body joint twists, and $\boldsymbol{S}(\boldsymbol{p})$ is the skew-symmetric matrix of the vector $\boldsymbol{p}$, so that $\boldsymbol{S}(\boldsymbol{p})\,\boldsymbol{R} = \boldsymbol{p} \times \boldsymbol{R}$. The base and end-effector geometric Jacobians can be found by applying spatial transformations to the spatial geometric Jacobian, as shown below. Refer to [7] for details on the spatial Jacobian and its related transformations.

[2]Sometimes called the body geometric Jacobian.

The base geometric Jacobian is

$$\boldsymbol{J}_g^0 = \begin{pmatrix} \mathbf{I} & S({}^E\boldsymbol{p}_0) \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \boldsymbol{J}_g^S, \tag{2.11}$$

and the end-effector geometric Jacobian is

$$\boldsymbol{J}_g^E = \boldsymbol{Ad}\left({}^E\boldsymbol{T}_0\right) \boldsymbol{J}_g^S, \tag{2.12}$$

which define the maps of (2.5) and (2.6).

The transformations applied to the spatial Jacobian in (2.11) and (2.12) are adjoint map transformation from the spatial coordinates to the base and end-effector coordinates, respectively. The former for simplicity is written out in matrix form as it involves no rotation.

Chapter 3

Underwater manipulator control

This chapter develops a control scheme for USM manipulation that compensates for the reaction forces and hydrostatic forces it experiences. The first section introduces the generalized Jacobian matrix—previously only applied to space robots—and applies it to the USM. The section ends with a discussion of the singularities of the GJM, focusing on the practical differences for a robot in space and a robot under water.

The second section discusses how hydrostatic forces affect the USM. The effect is different than on traditional underwater vehicles because the COM and COB are strongly dependent on the joint configuration. The locations of the COM and COB are derived as functions of the joint configuration and are used to predict the rotation between the vehicle's current orientation and its orientation at hydrostatic equilibrium. An estimate of the end-effector velocities caused by the hydrostatics, using measurements obtainable with a low-cost inertial measurement unit (IMU), is also derived. Simple velocity and position feedback laws are defined to correct the hydrostatics disturbance. A method for damping rotational oscillations, using the predicted equilibrium rotation, is also developed.

The third section compares some existing inverse kinematic methods for the present application, which is necessary to realize GJM-based control and combine it with the hydrostatics compensation. The fourth section presents some existing methods of joint limit avoidance, which is necessary to avoid mechanical damage and loss of control. An improvement upon the current methods is developed and proven valid. The fifth section handles the control allocation problem and generalizes control allocation to vehicles with arbitrary and time-variant thrust configuration.

The control approach here is general, like the model in Chapter 2, and can be applied to any robotic system that meets the assumptions of the model, that is, most robotic manipulators. Thrusters are required only for the optional oscillation damping in Section 3.2.7.

3.1 Compensating for reaction forces

This section summarizes the definition of the GJM from [41], which will be used to solve the inverse kinematics problem. The GJM is chosen because it makes it possible to control the end-effector relative to the inertial frame, despite the reaction forces created by the joint motion. Using the GJM compensates for both rotational and translational reaction forces. The virtual manipulator has been investigated as another alternative, but it does not compensate for rotational errors and would require active attitude control [25], rendering it unsuitable.

3.1.1 The generalized Jacobian matrix

The GJM, introduced in [5], [23], provides a map from the joint velocities to the end-effector velocity twist, expressed in the inertial frame:

$$ {}^{I}\boldsymbol{\nu}_{I,E} = \boldsymbol{J}^{*}(\boldsymbol{q}, {}^{I}\boldsymbol{R}_{0})\,\dot{\boldsymbol{q}}, \tag{3.1} $$

where $\boldsymbol{J}^{*}$ is the GJM. This map contrasts that of the fixed-base manipulator Jacobians defined in Section 2.2.1, which map to end-effector velocities expressed in, for instance, the base or end-effector frame. By mapping to velocities in an inertial frame, the GJM takes into account the displacement of the floating manipulator that occurs as a result of moving its joints. Figure 1.2 illustrates how base displacement can occur.

The GJM was originally developed for manipulators mounted on free-floating vehicles in space, typically satellites [24], which experience negligible external forces. For these systems, conservation of momentum applies. As described in Section 1.2, underwater manipulators are subject to external forces such as hydrostatic restoring forces and hydrodynamic effects. Hence, the assumption of no external forces does not hold for these systems. It has so far been unknown whether the GJM is valuable for systems that do experience some external forces. The aim of this chapter and the simulations is Chapter 4 is to investigate the usability of the GJM for USMs.

Below follows a complete definition of the GJM, adapted from [41]. The symbols used are:

- $\boldsymbol{q} \in \mathbb{R}^n$ are the joint variables,
- $\boldsymbol{r}_i \in \mathbb{R}^3$ is the position of the COM of link i,
- $\boldsymbol{r}_g \in \mathbb{R}^3$ is the position of the total COM,
- $\boldsymbol{r}_{0g} = \boldsymbol{r}_g - \boldsymbol{r}_0$ is a vector from the base to the total COM,
- $\boldsymbol{r}_{0i} = \boldsymbol{r}_i - \boldsymbol{r}_0$ is a vector from the base to the COM of link i,
- $\boldsymbol{p}_j \in \mathbb{R}^3$ is the position of joint j,
- $\boldsymbol{p}_r \in \mathbb{R}^3$ is the position of the end-effector,
- $\boldsymbol{p}_{0r} = \boldsymbol{p}_r - \boldsymbol{r}_0$ is a vector from the base to the end-effector,
- $\boldsymbol{k}_j \in \mathbb{R}^3$ is a unit vector indicating the rotation axis of joint j.
- ${}^I\boldsymbol{R}_0 \in \mathbb{R}^{3\times3}$ is the rotation from the inertial frame to the base,
- m_i is the mass of link i,
- m_{tot} is the total mass, and
- $\boldsymbol{I}_i \in \mathbb{R}^{3\times3}$ is the inertia tensor of link i,

where the vectors above are expressed in the inertial frame, and are illustrated for an example vehicle in Figure 3.1.

To define the GJM, it is necessary to first define the manipulator Jacobian[1]

$$\boldsymbol{J}_m = \begin{pmatrix} \boldsymbol{k}_1 \times (\boldsymbol{p}_r - \boldsymbol{p}_1) & \boldsymbol{k}_2 \times (\boldsymbol{p}_r - \boldsymbol{p}_2) & \dots & \boldsymbol{k}_n \times (\boldsymbol{p}_r - \boldsymbol{p}_n) \\ \boldsymbol{k}_1 & \boldsymbol{k}_2 & \dots & \boldsymbol{k}_n \end{pmatrix} \in \mathbb{R}^{6\times n}, \tag{3.2}$$

[1]Note that $\boldsymbol{J}_m$ is identical to the base geometric Jacobian $\boldsymbol{J}_{g,0}$ defined in Section 2.2.1.

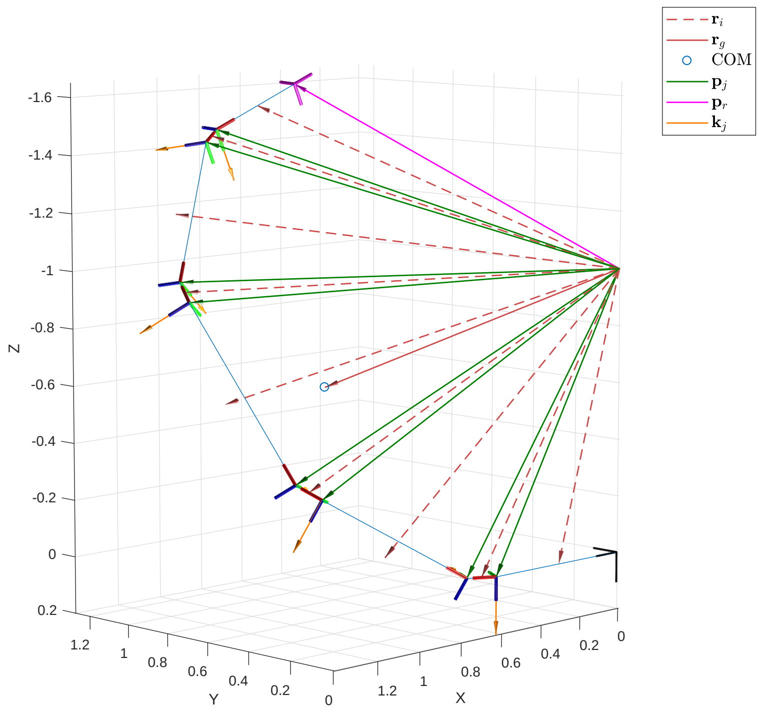


Figure 3.1: Visualization of the vectors that define the GJM, shown for an 8-link manipulator. The joint frames are shown as mutually perpendicular red/green/blue rods, connected by blue lines. The base frame is shown in black at the origin, and the end-effector frame is pink.

the base Jacobian

$$\boldsymbol{J}_s = \begin{pmatrix} \mathbf{I}_{3\times 3} & -\boldsymbol{S}(\boldsymbol{p}_{0r}) \\ \mathbf{0} & \mathbf{I}_{3\times 3} \end{pmatrix} \in \mathbb{R}^{6\times 6}, \tag{3.3}$$

and their extensions

$$\begin{aligned} \hat{\boldsymbol{J}}_s &= \boldsymbol{J}_s \begin{pmatrix} \boldsymbol{S}(\boldsymbol{r}_{0g}) \\ \mathbf{I}_{3\times 3} \end{pmatrix} \in \mathbb{R}^{6\times 3}, \\ \hat{\boldsymbol{J}}_m &= \boldsymbol{J}_m - \boldsymbol{J}_s \begin{pmatrix} \boldsymbol{J}_{Tw}/m_{\text{tot}} \\ \mathbf{0} \end{pmatrix} \in \mathbb{R}^{6\times n}, \end{aligned} \tag{3.4}$$

where $\hat{\boldsymbol{J}}_m$ happens to be identical to the Jacobian of the virtual manipulator [25]. The GJM can then be defined as

$$\boldsymbol{J}^* = \hat{\boldsymbol{J}}_m - \hat{\boldsymbol{J}}_s \boldsymbol{I}_s^{-1} \boldsymbol{I}_m \in \mathbb{R}^{6\times n}, \tag{3.5}$$

where

$$\boldsymbol{I}_s = \boldsymbol{I}_\omega + m_{\text{tot}} \boldsymbol{S}(\boldsymbol{r}_g \boldsymbol{r}_{0g}) \in \mathbb{R}^{3\times 3}, \tag{3.6}$$

$$\boldsymbol{I}_m = \boldsymbol{I}_\phi - \boldsymbol{S}(\boldsymbol{r}_g)\boldsymbol{J}_{Tw} \in \mathbb{R}^{3\times n}, \tag{3.7}$$

$$\boldsymbol{I}_\omega = \sum_{i=1}^{n} \left(\boldsymbol{I}_i - m_i \boldsymbol{S}(\boldsymbol{r}_i)\boldsymbol{S}(\boldsymbol{r}_{0i})\right) + \boldsymbol{I}_0, \tag{3.8}$$

$$\boldsymbol{I}_\phi = \sum_{i=1}^{n} \left(\boldsymbol{I}_i \boldsymbol{J}_{Ri} + m_i \boldsymbol{S}(\boldsymbol{r}_i)\boldsymbol{J}_{Ti}\right), \tag{3.9}$$

and

$$J_{Ti} = \begin{pmatrix} \boldsymbol{k}_1 \times (\boldsymbol{r}_i - \boldsymbol{p}_1) & \boldsymbol{k}_2 \times (\boldsymbol{r}_i - \boldsymbol{p}_2) & \dots & \boldsymbol{k}_i \times (\boldsymbol{r}_i - \boldsymbol{p}_i) & \boldsymbol{0} & \dots & \boldsymbol{0} \end{pmatrix} \in \mathbb{R}^{3\times n}, \tag{3.10}$$

$$J_{Ri} = \begin{pmatrix} \boldsymbol{k}_1 & \boldsymbol{k}_2 & \dots & \boldsymbol{k}_i & \boldsymbol{0} & \dots & \boldsymbol{0} \end{pmatrix} \in \mathbb{R}^{3\times n}, \tag{3.11}$$

$$J_{Tw} = \sum_{i=1}^{n} m_i J_{Ti} \in \mathbb{R}^{3\times n}. \tag{3.12}$$

Further details on the derivation of the GJM can be found in [5], [23], and more thoroughly in [41].

3.1.2 Singularities of the GJM

A manipulator is said to be in a singular configuration when its Jacobian is singular, or rank deficient. When the Jacobian is singular, some of its task-space velocities are impossible to reach, and its inverse mapping is not well-defined. For square singular Jacobians, the inverse matrix is undefined. In the neighborhood of a singularity, the task velocities may be theoretically reachable, but only with excessive joint velocities. Large joint velocities can cause spurious and unpredictable motion, and may exceed the physical ability of the joint actuators.

Nearly all manipulators have *kinematic singularities*, which are singularities due to its kinematic structure. They can occur at the workspace boundary, and at points where joint axes align. The GJM also exhibits *dynamic singularities*, which are dependent on the dynamic properties of the vehicle. Kinematic and dynamic singularities reflect physical limitations and are only avoidable by staying away from them, not by, for instance, expressing the problem differently.

A USM is subject to more significant reaction forces than typical free-floating space manipulators. For a normal USM, the base link is of approximately the same size and mass as the other links, while for a satellite manipulator (a typical application of GJM control) the base is usually significantly heavier than the entire manipulator. An example is the "Engineering Test Satellite VII", which is a 140 kg arm mounted on a 2550 kg base, used for in-orbit experiments [24]. The workspace of a free-floating USM is therefore fairly small in comparison to its kinematic dimensions. Singularities at workspace boundary and inside the workspace are rarely far away, and it is necessary to avoid them in a way that minimizes the resulting tracking errors. This will be addressed in Section 3.3.

3.2 Compensating for hydrostatic forces

The USM is assumed to be neutrally buoyant—it experiences no translational hydrostatic forces. However, due to its local hydrostatic restoring moments, each link will have a tendency to move toward its equilibrium orientation. When many links are connected, as is the case for a USM, the structure as a whole will experience restoring moments that depend on the joint configuration. This disturbance makes it harder to perform manipulation tasks. If not accounted for, the entire vehicle may rotate out of position, causing large end-effector velocity and position errors. The following describes the hydrostatic forces, and how to compensate for them.

The definition of the restoring force vector in Section 3.2.1 is a well established result, but normally assumes that the COM and COB are fixed in the body frame of the vehicle—a reasonable assumption for larger UVMSs, but not for USRs and USMs. Snake robots have configuration-dependent locations of the COM and COB, which are derived as functions of the joint positions in Section 3.2.2, to yield a more general expression for the restoring force vector. The expression is used to predict the orientation of the base before and after reaching hydrostatic equilibrium, and to define velocity and position feedback terms to counter the hydrostatics-induced rotation.

3.2.1 Quantifying the hydrostatic forces

By Assumption 6, the USM is neutrally buoyant, which means that the linear component of the restoring force vector disappears. If the mass distribution of the body is such that its total COM aligns with its total COB, the rotational forces also disappear, but this is not generally the case. The following will define the restoring force vector.

The weight of a submerged object is

$$W = gm_{\text{tot}}, \tag{3.13}$$

where m_{tot} is its mass and g is the gravitational acceleration [42]. By Archimedes' principle [43], the buoyancy of a submerged object is

$$B = \rho g \nabla_{\text{tot}}, \tag{3.14}$$

where ρ is the density of the surrounding fluid, and ∇_{tot} is the displaced volume of the object.

Gravity acts through the COM of the object. The COM can be found either experimentally; computationally with the help of a computer-aided design (CAD) model; or by calculating the weighted centroid of the mass distribution. For an object of uniform mass distribution, the COM is at its centroid.

Buoyancy acts through the COB. The COB is located at the COM of the displaced volume, meaning it would coincide with the COM if the object has a uniform mass distribution. The COM and COB of each link is known by Assumption 7.

It is simple to express the forces of gravity and buoyancy in a coordinate system defined with one axis straight up or down. The z-axis of the inertial frame $\mathcal{F}_I$ points down, and we get

$${}^{I}\boldsymbol{f}_{\text{grav}} = \begin{pmatrix} 0 \\ 0 \\ W \end{pmatrix}, \qquad {}^{I}\boldsymbol{f}_{\text{buoy}} = -\begin{pmatrix} 0 \\ 0 \\ B \end{pmatrix}, \tag{3.15}$$

illustrated in Figure 3.2.

Having defined the forces of gravity and buoyancy, the restoring force vector expressed in the base frame is [6]

$${}^{0}\boldsymbol{g}({}^{I}\boldsymbol{R}_0, \boldsymbol{q}) = -\begin{pmatrix} {}^{I}\boldsymbol{R}_0 \left({}^{I}\boldsymbol{f}_{\text{grav}} + {}^{I}\boldsymbol{f}_{\text{buoy}} \right) \\ \boldsymbol{S}({}^{0}\boldsymbol{p}_{\text{cm}})\, {}^{I}\boldsymbol{R}_0\, {}^{I}\boldsymbol{f}_{\text{grav}} + \boldsymbol{S}({}^{0}\boldsymbol{p}_{\text{cb}})\, {}^{I}\boldsymbol{R}_0\, {}^{I}\boldsymbol{f}_{\text{buoy}} \end{pmatrix}. \tag{3.16}$$

It is clear from (3.15) and (3.16) that when $W = B$, the upper three (translative) elements of ${}^{0}\boldsymbol{g}$ vanish, while the lower three (rotational) elements can still be nonzero, given that ${}^{0}\boldsymbol{p}_{\text{cm}} \neq {}^{0}\boldsymbol{p}_{\text{cb}}$.

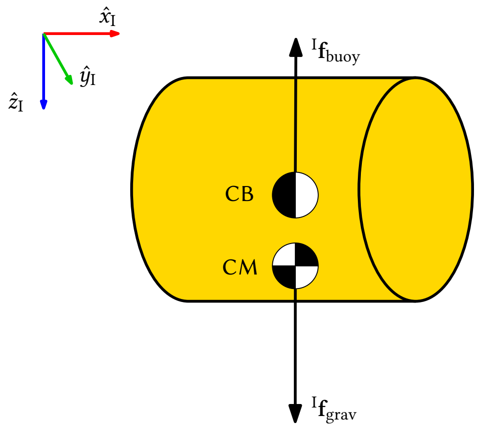

$\hat{x}_\mathrm{I}$
$\hat{y}_\mathrm{I}$
$\hat{z}_\mathrm{I}$
${}^\mathrm{I}\mathbf{f}_\mathrm{buoy}$
CB
CM
${}^\mathrm{I}\mathbf{f}_\mathrm{grav}$

Figure 3.2: Forces of gravity and buoyancy on a submerged body, shown in its equilibrium position. If the magnitudes of the forces are equal, the body is neutrally buoyant and will not sink or float. If the COM and COB have different locations, the body will experience hydrostatic restoring moments.

3.2.2 Finding the locations of the COM and COB

The location of the total COM of the USM is dependent on its configuration and can be calculated as a weighted sum of the positions of each link's COM [25]. Expressed in the base frame:

$$ {}^0\boldsymbol{p}_\mathrm{cm}(\boldsymbol{q}) = \sum_{i=1}^{n} {}^0\boldsymbol{p}_{\mathrm{cm},\,i}(\boldsymbol{q}) \frac{m_i}{m_\mathrm{tot}} \tag{3.17} $$

where

$$ {}^0\boldsymbol{p}_{\mathrm{cm},\,i}(\boldsymbol{q}) = {}^0\boldsymbol{R}_i(\boldsymbol{q})\, {}^i\boldsymbol{p}_{\mathrm{cm},\,i} + {}^0\boldsymbol{p}_i(\boldsymbol{q}) \tag{3.18} $$

defines the position of the COM of link i in the base frame.

The method can be repeated for buoyancy: The COB of a link is at its geometric center. Mirroring the equations above, a weighted sum of the locations of each link's COB can express the location of the overall COB. However, this time, the weights are not based on the mass of the links, but on their volumes:

$$ {}^0\boldsymbol{p}_\mathrm{cb}(\boldsymbol{q}) = \sum_{i=1}^{n} {}^0\boldsymbol{p}_{\mathrm{cb},\,i}(\boldsymbol{q}) \frac{\nabla_i}{\nabla_\mathrm{tot}} \tag{3.19} $$

where

$$ {}^{0}\boldsymbol{p}_{\text{cb},i}(\boldsymbol{q}) = {}^{0}\boldsymbol{R}_i(\boldsymbol{q})\, {}^{i}\boldsymbol{p}_{\text{cb},i} + {}^{0}\boldsymbol{p}_i(\boldsymbol{q}) \tag{3.20} $$

as before expresses a link's COB in the base frame as a function of the joint configuration.

3.2.3 Rotation caused by hydrostatic forces

The aim of this section is to find the rotation of the base link before and after reaching hydrostatic equilibrium. In an attitude controller, the rotation of the base at equilibrium can be used as the setpoint, to aid the vehicle in reaching its hydrostatic equilibrium faster and with less oscillations.

The end-effector position in the inertial frame is ${}^{I}\boldsymbol{p}_E$. If the joints and thrusters are stationary, and the COM does not move considerably, then the most significant forces affecting the USM are the hydrostatic forces. By calculating how much, and around what axis the hydrostatic forces causes the USM to move, it is possible to estimate the final orientation of the USM.

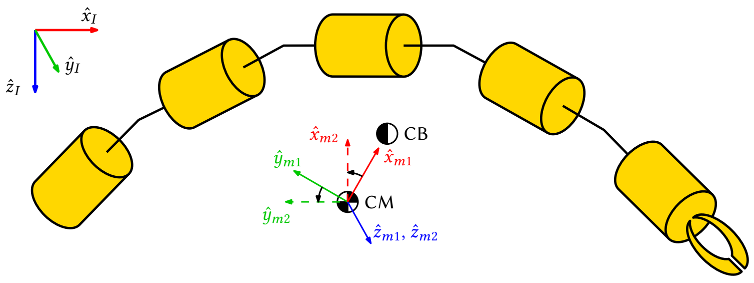


Figure 3.3: Frames $\mathcal{F}_{m1}$ and $\mathcal{F}_{m2}$ in relation to the COM, COB, and the inertial frame. These frames represent the relative locations of the COM and COB before and after the vehicle reaches hydrostatic equilibrium. The origins of both frames are at the COM. Frame $\mathcal{F}_{m1}$ is shown with solid arrows and $\mathcal{F}_{m2}$ with dashed arrows.

To do so, define two new coordinate frames $\mathcal{F}_{m1}$ and $\mathcal{F}_{m2}$, shown in Figure 3.3, both with origins at the COM. $\mathcal{F}_{m1}$ represents the relative locations of the COM and COB before hydrostatic equilibrium, while $\mathcal{F}_{m2}$ represents their locations at equilibrium—when the COB is directly above the COM. The x-axis of $\mathcal{F}_{m1}$ is defined parallel to the vector from the COM to the COB, ${}^{I}\boldsymbol{r}_{\text{cmcb}} = {}^{I}\boldsymbol{p}_{\text{cb}} - {}^{I}\boldsymbol{p}_{\text{cm}}$. The orthogonal, unnormalized basis vectors for $\mathcal{F}_{m1}$ and $\mathcal{F}_{m2}$, in the inertial frame, are

$$\begin{aligned} {}^I\hat{\boldsymbol{x}}'_{m1} &= {}^I\boldsymbol{r}_{\text{cm,cb}} \\ {}^I\hat{\boldsymbol{y}}'_{m1} &= {}^I\hat{\boldsymbol{z}}'_{m1} \times {}^I\hat{\boldsymbol{x}}'_{m1} \\ {}^I\hat{\boldsymbol{z}}'_{m1} &= \begin{cases} {}^I\boldsymbol{f}_{\text{grav}} \times {}^I\boldsymbol{r}_{\text{cm,cb}} & \text{when } {}^I\boldsymbol{r}_{\text{cm,cb}} \nparallel {}^I\boldsymbol{f}_{\text{grav}} \\ (1,0,0)^{\mathrm{T}} & \text{otherwise,} \end{cases} \end{aligned} \tag{3.21}$$

and

$$\begin{aligned} {}^I\hat{\boldsymbol{x}}'_{m2} &= (0,0,-1)^{\mathrm{T}} \\ {}^I\hat{\boldsymbol{y}}'_{m2} &= {}^I\hat{\boldsymbol{z}}'_{m2} \times {}^I\hat{\boldsymbol{x}}'_{m2} \\ {}^I\hat{\boldsymbol{z}}'_{m2} &= {}^I\hat{\boldsymbol{z}}'_{m1}, \end{aligned} \tag{3.22}$$

where ${}^I\boldsymbol{f}_{\text{grav}}$ is the downward-pointing gravity vector. Note the split definition of ${}^I\hat{\boldsymbol{z}}'_{m1}$ in (3.21): The first case is ambiguous at equilibrium[2], i.e. when ${}^I\boldsymbol{r}_{\text{cm,cb}}$ is parallel to the gravity vector. This is resolved by setting ${}^I\hat{\boldsymbol{z}}'_{m1}$ equal to an arbitrary horizontal vector.

By denoting the unit length counterparts of the basis vectors using the same notation sans the prime mark ($'$), we can define rotation matrices that describe the orientations of $\mathcal{F}_{m1}$ and $\mathcal{F}_{m2}$ relative to the inertial frame:

$$\begin{aligned} {}^I\boldsymbol{R}_{m1} &= \left({}^I\hat{\boldsymbol{x}}_{m1}, {}^I\hat{\boldsymbol{y}}_{m1}, {}^I\hat{\boldsymbol{z}}_{m1}\right) \\ {}^I\boldsymbol{R}_{m2} &= \left({}^I\hat{\boldsymbol{x}}_{m2}, {}^I\hat{\boldsymbol{y}}_{m2}, {}^I\hat{\boldsymbol{z}}_{m2}\right). \end{aligned} \tag{3.23}$$

Knowing these rotations makes it possible to determine the rotation from the current orientation of any given link to the orientation of that link at hydrostatic equilibrium. Typically, the base link or end-effector link will be most interesting, and the following derivation is valid for the base link rotation.

The current rotation of the base link is ${}^I\boldsymbol{R}_0$, and the rotation of the base link at hydrostatic equilibrium is denoted ${}^I\boldsymbol{R}_{0,\text{eq}}$. We assume that the equilibrium orientation is the orientation closest to the current orientation while also being at equilibrium, given the current joint configuration. Using the $\mathcal{F}_{m1}$ and $\mathcal{F}_{m2}$ frames and ${}^{m1}\boldsymbol{R}_{m2} = {}^{m1}\boldsymbol{R}_I\, {}^{m2}\boldsymbol{R}_I^{-1}$,

[2]It is strictly speaking ambiguous at *both* equilibria: The stable equilibrium when the vehicle is upright, and the unstable equilibrium when the vehicle is upside-down. This has no impact on the solution.

the equilibrium orientation can be found as

$$ {}^{I}\boldsymbol{R}_{0,\mathrm{eq}} = {}^{I}\boldsymbol{R}_{0}\, {}^{0}\boldsymbol{R}_{m1}\, {}^{m1}\boldsymbol{R}_{m2}\, {}^{0}\boldsymbol{R}_{m1}^{-1}. \tag{3.24} $$

In other words, rotate first from the inertial frame to the current base frame, then from the base frame to the "before equilibrium COM frame" $\mathcal{F}_{m1}$, then complete the predicted hydrostatics-rotation ${}^{m1}\boldsymbol{R}_{m2}$ itself, and finally, rotate back to the new base frame with ${}^{0}\boldsymbol{R}_{m1}^{-1} = {}^{m1}\boldsymbol{R}_{0}$.

3.2.4 Velocities caused by hydrostatic forces

It is also possible to estimate the velocities of the end-effector based on the rotation of an arbitrary frame attached to the USM. If we assume small linear external forces, in particular, small linear components of the hydrodynamic forces, the USM will rotate about the COM during pure hydrostatic restoring motion. As the USM is rigid, the angular velocity of a frame attached to the COM will be the same as the angular velocity of any parallel frame on the USM. We have the velocity of the end-effector

$$ {}^{I}\boldsymbol{\omega}_{E} = {}^{E}\boldsymbol{R}_{0}\, {}^{0}\boldsymbol{R}_{m1}\, {}^{I}\boldsymbol{\omega}_{m1}, \tag{3.25} $$

and use it to predict its linear velocity, given pure rotation around the COM, as

$$ {}^{I}\boldsymbol{v}_{E} = S({}^{E}\boldsymbol{p}_{cm})\, {}^{I}\boldsymbol{\omega}_{E}. \tag{3.26} $$

The above provides a rudimentary estimate of the linear end-effector velocity which can be used instead of a direct measurement. A direct measurement would require complex and expensive sensor equipment, such as a Doppler velocity log (DVL), and is therefore not necessarily available.

3.2.5 Velocity feedback

Velocity estimates are often available in some form, such as the estimates in Section 3.2.4, and this allows for velocity feedback. Angular velocity can be measured directly with a gyroscope, which is available for instance as part of an IMU. Linear velocity can also be estimated from IMU accelerometer and gyroscope data through integration and sensor fusion, with the drawback that the estimate drifts over time, depending on the accuracy of the IMU [6]. One can produce better estimates by including a DVL, which measures velocity relative to the sea floor through Doppler effects [44].

When end-effector velocity measurements or estimates are available, it is possible to use velocity feedback for end-effector control in the face of disturbances. While designing a potentially complex velocity controllers is worth considering, it is outside the scope of this thesis, and a simple controller is used here for illustration and comparison. The controller used simply adds the measured velocity error to the current velocity command:

$$\dot{\boldsymbol{t}}_c = \dot{\boldsymbol{t}}_d + \left(\dot{\boldsymbol{t}}_d - \dot{\boldsymbol{t}}_m\right) \tag{3.27}$$

where the subscripts are "c" for commanded value, "d" for the desired value, and "m" for the measured or estimated value. This control algorithm is essentially an error P controller with unit gain, where the control output $(\dot{\boldsymbol{t}}_d - \dot{\boldsymbol{t}}_m)$ is added onto the original control signal $\dot{\boldsymbol{t}}_d$. A slightly more ornate option could be to use, for instance, the real-time motion compensation algorithm of [45].

3.2.6 Position feedback

Estimates of end-effector position are harder to obtain than estimates of velocity. The position is possible to estimate by dead-reckoning, which combines and integrates velocity and acceleration level measurements. Due to the numerical integration, the position estimates will drift. If only acceleration measurements are available, the drift error will be quadratic in time (due to the double integration), and with velocity measurements, it will be linear in time (due to the single integration) [46]. Estimators based on double integration are usually not accurate for a long enough time to be useful for precision tasks. For instance, the USAF SNU84-1 categorizes a "precision inertial navigation system (INS)" as having a position error growth rate of less than $0.5\,\mathrm{M/h} \approx 25\,\mathrm{cm/s}$ [47].[3] For this reason, DVL measurements are normally used in INSs [48]. These indirect estimation methods are not perfect, but are often the only available methods for ocean operations away from human-made structures.

Various methods exist for direct position measurement underwater. While global navigation satellite systems (GNSSs) are unavailable underwater, a similar method based

[3]M/h refers to nautical miles per hour.

on triangulation with ranges to acoustic transponders, called long baseline (LBL), is possible. Another possibility is ultra-short baseline (USBL), using a sonar array for position measurements [48]. The drawback of all underwater position measurement methods is that the sensor equipment must be installed on site—sensors on board the vehicle are not sufficient for most methods. An exception is camera-based simultaneous localization and mapping (SLAM), which is a viable method in situations with good visibility and does not require equipment installed in the environment [49].

A simple method for incorporating position data into the manipulator control scheme is to use closed-loop inverse kinematics (CLIK) [50]. CLIK was developed to remove the accumulated integration error from joint trajectory reconstruction [51], but is here used to also account for position error that occurs during underwater GJM-based manipulation. A first-order CLIK algorithm is [51]

$$\dot{\boldsymbol{q}}_c = \boldsymbol{J}^\dagger(\boldsymbol{q}) \left(\dot{\boldsymbol{t}}_d + \boldsymbol{K}_{\text{clik}} \left[\boldsymbol{t}_d - \boldsymbol{t}_m\right]\right) + \left(\mathrm{I} - \boldsymbol{J}^\dagger(\boldsymbol{q})\boldsymbol{J}(\boldsymbol{q})\right) \dot{\boldsymbol{q}}_0, \tag{3.28}$$

where the subscripts are as in Section 3.2.5, and $\boldsymbol{K}_{\text{clik}}$ is a constant, positive definite gain matrix. The last term allows exploitation of the redundancy, with the arbitrary joint velocity vector $\dot{\boldsymbol{q}}_0$. In this thesis, the inverse kinematics in CLIK are removed, so that (3.28) reduces to

$$\dot{\boldsymbol{t}}_c = \dot{\boldsymbol{t}}_d + \boldsymbol{K}_{\text{clik}} \left(\boldsymbol{t}_d - \boldsymbol{t}_m\right). \tag{3.29}$$

The advantage is that (3.29) can be used with any inverse kinematic method, some of which are discussed below. Note again that this is a very simple position feedback control scheme. It is likely possible to achieve better performance with other methods, but the aim here is to illustrate the potential benefits of closed-loop control with position feedback compared to control without position feedback—not to investigate the feedback algorithms thoroughly.

3.2.7 Damping hydrostatics-induced oscillations

If the USM is not at its hydrostatic equilibrium, it will return to the equilibrium by itself. Although water creates much more damping than air, the dynamics of the rotational motion toward equilibrium are qualitatively similar to the oscillations of a pendulum: It will swing past its equilibrium point, turn around, swing past the equilibrium in the opposite direction, and so on for some time before coming to a halt. While the feedback algorithms of Sections 3.2.5 and 3.2.6 help stabilize end-effector at its correct position though joint motion, the accuracy will likely drop if the algorithm needs to compensate for oscillations as well. Power consumption will also increase. To alleviate accuracy and energy concerns, the thrusters can be used to damp the oscillations, by activating a setpoint attitude controller immediately after a completed task velocity trajectory. The attitude controller must estimate the equilibrium orientation as a function of the current orientation and joint configuration, and use that as its setpoint. This is done in (3.24).

The orientation ${}^{I}\boldsymbol{R}_{0,\mathrm{eq}}$ can then be used as the setpoint for an attitude controller. A suitable controller for this is the quaternion-based proportional-derivative (PD) controller of [52]. While technically a full pose (position and orientation) controller, the position and orientation are decoupled, making it trivial to adapt the controller to the present requirements by extracting the attitude “sub-controller”.

The controller uses unit quaternions for orientation representation, which makes it singularity free. A controlled based on Euler angles would experience the gimbal lock singularity when yaw and roll axes align [36]. Most 3D geometry software packages, including the Robotics System Toolbox in MATLAB [53], include methods for conversion between unit quaternions and rotation matrices. Below,

$$\boldsymbol{h} = \begin{pmatrix} \eta \\ \boldsymbol{\epsilon} \end{pmatrix} \in \mathcal{H} \tag{3.30}$$

is the unit quaternion for the current orientation,

$$\boldsymbol{h}_d = \begin{pmatrix} \eta_d \\ \boldsymbol{\epsilon}_d \end{pmatrix} \in \mathcal{H} \tag{3.31}$$

is the desired orientation, and

$$\tilde{\boldsymbol{h}} = \begin{pmatrix} \tilde{\eta} \\ \tilde{\boldsymbol{\epsilon}} \end{pmatrix} = \overline{\boldsymbol{h}}_d \boldsymbol{h} = \begin{pmatrix} \eta_d & \boldsymbol{\epsilon}_d^{\mathrm{T}} \\ -\boldsymbol{\epsilon}_d & \eta_d \mathrm{I}_{3\times 3} - S(\boldsymbol{\epsilon}_d) \end{pmatrix} \begin{pmatrix} \eta \\ \boldsymbol{\epsilon} \end{pmatrix} \in \mathcal{H}, \tag{3.32}$$

represents the quaternion error, or difference between $\boldsymbol{h}$ and $\boldsymbol{h}_d$. The set $\mathcal{H}$ is the set of all unit quaternions.

With the position control part removed, and notation changed to match this thesis, the control law is

$$\boldsymbol{m} = -\boldsymbol{K}_d\, \boldsymbol{\omega} - {}^{I}\boldsymbol{R}_0^{\mathrm{T}} \boldsymbol{K}_p \operatorname{sgn}(\tilde{\eta})\tilde{\boldsymbol{\epsilon}} + \boldsymbol{g}_{\mathrm{rot}}, \tag{3.33}$$

where $\boldsymbol{m}$ are the commanded moments to be acted on the vehicle, $\boldsymbol{K}_d = \boldsymbol{K}_d^{\mathrm{T}} \succ \boldsymbol{0}$ is a derivative gain matrix, $\boldsymbol{K}_p = \boldsymbol{K}_p^{\mathrm{T}} \succ \boldsymbol{0}$ is a proportional gain matrix, $\operatorname{sgn}(\cdot)$ is the signum operator defined as

$$\operatorname{sgn}(x) = \begin{cases} -1 & \text{when } x < 0 \\ 1 & \text{otherwise} \end{cases}, \tag{3.34}$$

and $\boldsymbol{g}_{\mathrm{rot}} \in \mathbb{R}^3$ is the angular component of the restoring forces vector.

Perfect attitude regulation corresponds to an orientation error ${}^{I}\tilde{\boldsymbol{R}}_0 = \mathbf{I}_{3\times3}$, equivalent to $\tilde{\boldsymbol{h}} = (\pm1, 0, 0, 0)^{\mathrm{T}}$, which in turn is equivalent to $\tilde{\eta} = \pm1$, because of the unit norm of the quaternion. The controller is shown to have $\tilde{\eta} = \pm1$ as an asymptotically stable equilibrium point, while also exhibiting no unstable equilibrium points or singular points [52] (under the assumption that $\boldsymbol{m}$ is perfectly realizable). These properties make it suitable for the task at hand, but care must be taken to account for possible ill-conditioning or loss of rank of the thrust configuration matrix, and the limited bandwidth of the thrusters. In this thesis, the thrust configuration matrix is assumed well-conditioned by Assumption 5.

3.3 Inverse kinematics

The methods in Sections 3.1 and 3.2 create a task-space velocity reference $\dot{\boldsymbol{t}}$ (with the exception of Section 3.2.7, which gives a vector of moments to be acted out by the thrusters). A technique for calculating joint space commands $\dot{\boldsymbol{q}}$ as a function of task space references $\dot{\boldsymbol{t}}$ is needed. Such methods are termed inverse kinematic methods. This section will discuss some existing inverse kinematic methods for USM control.

In general, the map between joint velocities $\dot{\boldsymbol{q}}$ task space velocities $\dot{\boldsymbol{t}}$ is expressed

$$\dot{\boldsymbol{t}} = \boldsymbol{J}\dot{\boldsymbol{q}}, \tag{3.35}$$

where $\boldsymbol{J}$ is the Jacobian matrix for the given task. The following will compare some existing methods for solving the inverse problem of (3.35).

3.3.1 The Jacobian pseudoinverse

Pseudoinversion of the Jacobian is one of the simplest methods of inverse kinematics. Following from the properties of the Moore-Penrose matrix pseudoinverse [54], a solution of the inverse kinematics is

$$\dot{\boldsymbol{q}} = \underbrace{\boldsymbol{J}^{\mathrm{T}}\left(\boldsymbol{J}\boldsymbol{J}^{\mathrm{T}}\right)^{-1}}_{\boldsymbol{J}^{\dagger}}\dot{\boldsymbol{t}}, \tag{3.36}$$

where $\boldsymbol{J}^{\dagger}$ is the Jacobian pseudoinverse. It is interesting to note, especially for comparison with other inverse kinematic methods, that (3.36) is the solution of the optimization problem [6]

$$\begin{aligned} \min_{\dot{\boldsymbol{q}}} \quad & \dot{\boldsymbol{q}}^{\mathrm{T}}\dot{\boldsymbol{q}} \\ \text{s.t.} \quad & \dot{\boldsymbol{t}} - \boldsymbol{J}\dot{\boldsymbol{q}} = \boldsymbol{0}. \end{aligned} \tag{3.37}$$

The optimization problem reveals that $\boldsymbol{J}^{\dagger}$ minimizes the squared sum of the joint velocities, while not allowing any deviation from the desired task space velocity. The main drawback is that the pseudoinverse does not avoid singularities of the Jacobian, which is unacceptable for a manipulator control system because the Jacobian is almost guaranteed to exhibit singularities.

It is also possible to weight the joint velocities. Static weights can be used to prioritize between different types of joints, and dynamic weights can offer joint limit avoidance (discussed in Section 3.4), among others. The weighted variant solves [6]

$$\begin{aligned} \min_{\dot{\boldsymbol{q}}} \quad & \dot{\boldsymbol{q}}^{\mathrm{T}}\boldsymbol{W}\dot{\boldsymbol{q}} \\ \text{s.t.} \quad & \dot{\boldsymbol{t}} - \boldsymbol{J}\dot{\boldsymbol{q}} = \boldsymbol{0} \end{aligned} \tag{3.38}$$

as

$$\dot{\boldsymbol{q}} = \underbrace{\boldsymbol{W}^{-1}\boldsymbol{J}^{\mathrm{T}}\left(\boldsymbol{J}\boldsymbol{W}^{-1}\boldsymbol{J}^{\mathrm{T}}\right)^{-1}}_{\boldsymbol{J}_{w}^{\dagger}}\dot{\boldsymbol{t}}, \tag{3.39}$$

where

$$\boldsymbol{W} = \mathrm{diag}(w_1, w_2, \ldots, w_n) \tag{3.40}$$

is a diagonal weight matrix.

3.3.2 Damped least-squares

Damped least-squares, introduced in [18], is a way of solving the inverse kinematics problem that avoids singularities of the Jacobian. It is the solution of the optimization problem

$$\min_{\dot{\boldsymbol{q}}} \quad \left\|\dot{\boldsymbol{t}} - \boldsymbol{J}\dot{\boldsymbol{q}}\right\|^2 + \lambda_d^2\|\dot{\boldsymbol{q}}\|^2, \tag{3.41}$$

which is

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\mathrm{T}}\left(\boldsymbol{J}\boldsymbol{J}^{\mathrm{T}} + \lambda_d^2\mathbf{I}\right)^{-1}\dot{\boldsymbol{t}}. \tag{3.42}$$

Here, perfect task velocity tracking is no longer an equality constraint when compared to the pseudoinverse solution. Instead, it is a term in the objective function which is weighted against a damping term with the damping factor λ_d. The damping term is designed to reduce joint velocities when the manipulator is near a singular configuration, which can be achieved by choosing λ_d^2 according to [55]

$$\lambda_d^2 = \begin{cases} 0 & \text{when } \sigma_{\min} \geq \epsilon \\ \left[1 - \left(\frac{\sigma_{\min}}{\epsilon}\right)^2\right]\lambda_{d,\max}^2 & \text{otherwise,} \end{cases} \tag{3.43}$$

where $\sigma_{\min}$ is the smallest singular value (which is a measure of the proximity to a singularity [56]), ϵ is the singular value threshold below which the damping becomes active, and $\lambda_{d,\max}$ is the maximum damping factor. With (3.43), the solution is a pure task velocity error minimization when the manipulator is sufficiently far from any singularities. As it approaches a singular configuration, the joint velocity damping term will dominate.

The damped least-squares method can also be modified with joint velocity weights [18]. The optimization problem then becomes

$$\min_{\dot{\boldsymbol{q}}} \quad \left\|\dot{\boldsymbol{t}} - \boldsymbol{J}\dot{\boldsymbol{q}}\right\|^2 + \lambda_d^2\|\dot{\boldsymbol{q}}\|^2 + \lambda_w^2\dot{\boldsymbol{q}}^{\mathrm{T}}\boldsymbol{W}\dot{\boldsymbol{q}}, \tag{3.44}$$

which is solved as

$$\dot{\boldsymbol{q}} = \left(\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J} + \lambda_d^2\mathbf{I} + \lambda_w^2\boldsymbol{W}\right)^{-1}\dot{\boldsymbol{t}}, \tag{3.45}$$

where λ_w must be balanced against λ_d to decide the relative influence of the standard damping term and the joint velocity weight term.

3.3.3 Numerical filtering

A shortcoming of the damped least-squares method is that it damps all velocity components equally, even though typically only one or a few of the components are singular. Numerical filtering separates the poorly-conditioned and the well-conditioned components, and damps only those that are nearly singular. All well-conditioned components are allowed to behave normally. The result is singularity avoidance that does not cause unnecessary errors in the task space velocities. In other words, numerical filtering achieves "the physically realizable limits of manipulator performance for the given task" [19, p. 551].

The inverse kinematic solution using numerical filtering can be written [51]

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\mathrm{T}}\left(\boldsymbol{J}\boldsymbol{J}^{\mathrm{T}} + \lambda_d^2\mathbf{I} + \lambda_n^2\sum_{i=k+1}^{m}\tilde{\boldsymbol{u}}_i\tilde{\boldsymbol{u}}_i^{\mathrm{T}}\right)^{-1}\dot{\boldsymbol{t}}, \tag{3.46}$$

where m is the total number of task velocity components so that $\dot{\boldsymbol{t}} \in \mathbb{R}^m$, k is the number of well conditioned components, and $\tilde{\boldsymbol{u}}_i$ is an estimate of the input singular vector $\boldsymbol{u}_i$ in the singular value decomposition (SVD)

$$\boldsymbol{J} = \boldsymbol{U}\Sigma\boldsymbol{V}^{\mathrm{T}} = \sum_{i=1}^{\min(m,n)}\sigma_i\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}}, \tag{3.47}$$

for an m by n Jacobian. The scaling factor λ_n must be balanced against λ_d, typically with $\lambda_n \gg \lambda_d$.

In 1988, Maciejewski and Klein wrote "… the SVD is in general too computationally expensive for use in real-time control", and they opted instead for estimates of the singular values [19, p. 532]. Because the estimates may be inaccurate, another damping term $\lambda_d^2\mathbf{I}$ in (3.46) is added as a "fallback" to assure singularity avoidance even if the numerically filtered damping fails. However, the computational burden of an SVD today is much smaller: The SVD of a typical USM Jacobian, performed with the command `[U,S,V] = svd(J, 'econ')` in MATLAB, on a modern personal computer[4], takes on the order of tens of microseconds, as shown in Figure 3.4. As underwater control systems can run at fairly low frequencies due to the slow dynamics, there is ample time for SVD. Compare, for instance, the Intel 80386 processors from 1988 to the Intel Kaby Lake processors from 2017 for quantitative differences in computing power.

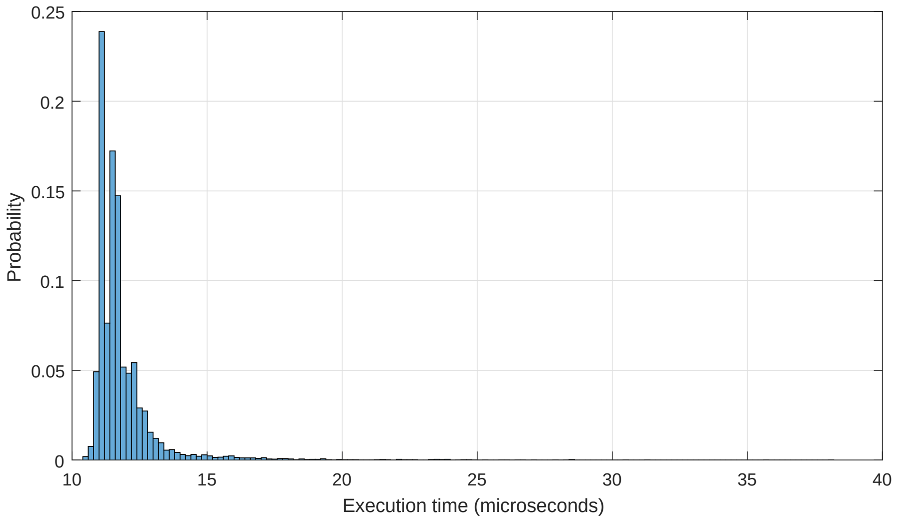


Figure 3.4: Execution times for an SVD of random-configuration GJMs of size 6-by-8, using MATLAB, on a typical personal computer.

3.4 Joint limit avoidance

When using a weighted redundancy resolution method, such as in Sections 3.3.1 and 3.3.2 above, the weights can be used to avoid joint limits. The weights of the diagonal weighting matrix in (3.40) are defined so that they increase when the corresponding joint approaches its limits and decreases when the joint approaches its midpoint. If any joints are close to their limits, the inverse kinematics method will prioritize using other joints. Joint limit avoidance is important for two main reasons: Reaching the limits can

[4]In this case on a Intel i7-7700HQ CPU.

cause mechanical damage to the joints and their actuators, and it will produce infeasible joint trajectories, which causes tracking errors in the task velocities.

A weighted least-norm solution is suggested in [57]. The authors use a performance criterion

$$H(\boldsymbol{q}) = \sum_{i=1}^{n} \frac{1}{4} \frac{(q_{i,\max} - q_{i,\min})^2}{(q_{i,\max} - q_i)(q_i - q_{q,\min})} \tag{3.48}$$

and its partial derivatives

$$\frac{\partial H(q_i)}{\partial q_i} = \frac{(q_{i,\max} - q_{i,\min})^2 (2q_i - q_{i,\max} - q_{i,\min})}{4(q_{i,\max} - q_i)^2 (q_i - q_{i,\min})^2} \tag{3.49}$$

to define the weights

$$w_i = \begin{cases} 1 + \left|\frac{\partial H}{\partial q_i}\right| & \text{when } \Delta\left|\frac{\partial H}{\partial q_i}\right| \geq 0 \\ 1 & \text{otherwise.} \end{cases} \tag{3.50}$$

These weights are exactly 1 at the midpoint, and approach infinity at the joint limits, so that joints in the middle of their range behave as in the unweighted case, while for joints approaching their limits, the joint velocities will decrease toward zero. The split definition in (3.50) has the advantage of only penalizing velocities of joints that approach their limits, not joints that retreat from them.

3.4.1 Weight normalization

The performance criterion $H(\boldsymbol{q})$ in (3.48), is normalized with respect to the range of the joints. However, the normalization does not carry over to its partial derivative, and as a result, the weights themselves are not normalized. As Figure 3.5 shows, this results in different weighting characteristics for different joint ranges. Even for joints that have the same physical range, the choice between radians or degrees as the joint angle unit dramatically changes the weighting.

To normalize the partial derivative of H, we require

$$\frac{\partial H}{\partial q_i} = f(t), \qquad \text{for } q_i = q_{i,\min} + t(q_{i,\max} - q_{i,\min}), \tag{3.51}$$

where $0 < t < 1$ is a parametrization of the position of q_i between its limits ($t = 0$ being the lower limit and $t = 1$ being the upper). If $\frac{\partial H}{\partial q_i}$ is a function of t only, then the function is independent of the joint range. The requirement of exclusive dependence on t is met by

$$\frac{\partial H}{\partial q_i} = \alpha \frac{(q_{i,\max} - q_{i,\min})^3 (2q_i - q_{i,\max} - q_{i,\min})}{(q_i - q_{i,\max})^2 (q_i - q_{i,\min})^2} \tag{3.52}$$

where α is an arbitrary scaling factor. Denoting the joint ranges as $q_{R,i} \triangleq q_{i,\max} - q_{i,\min}$ for brevity, and inserting $q_i = q_{i,\min} + t \cdot q_{ri}$, we can prove that (3.52) is independent of joint range:

$$\begin{aligned}
\frac{\partial H}{\partial q_i} &= \alpha \frac{q_{R,i}^3 \left(2(q_{i,\min} + tq_{R,i}) - q_{i,\max} - q_{i,\min}\right)}{\left(q_{i,\min} + tq_{R,i} - q_{i,\max}\right)^2 \left(q_{i,\min} + tq_{R,i} - q_{i,\min}\right)^2} \\
&= \alpha \frac{q_{R,i}^3 \left(2tq_{R,i} + q_{i,\min} - q_{i,\max}\right)}{\left(tq_{R,i} - q_{R,i}\right)^2 \left(tq_{R,i}\right)^2} \\
&= \alpha \frac{q_{R,i}^4 (2t - 1)}{q_{R,i}^4 (1 - t)^2 t^2} \\
&= \alpha \frac{2t - 1}{(1 - t)^2 t^2} \\
&= f(t) \quad \square
\end{aligned} \tag{3.53}$$

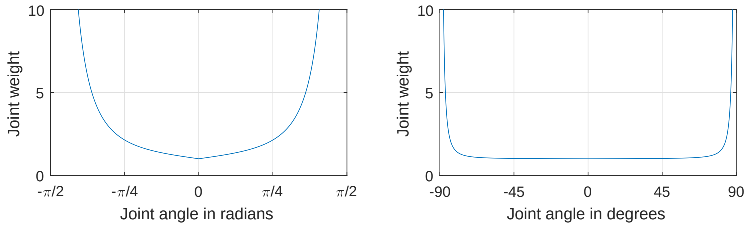


Figure 3.5: Weight characteristics for different angle units, using unnormalized weights.

Combining (3.50) with (3.52) gives the weight characteristics shown in Figure 3.6 (using $\alpha = 0.001$). It is evident that the weight characteristics are indeed independent of the joint range.

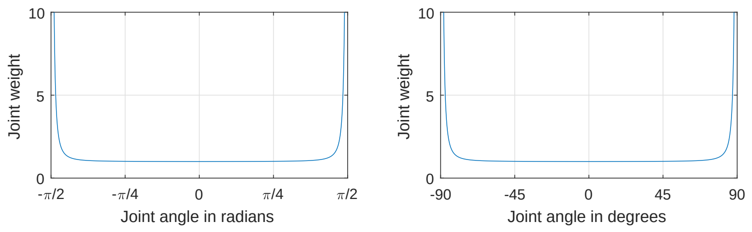


Figure 3.6: Weight characteristics for different angle units, using normalized weights.

3.4.2 Weight offset

Denote the weight matrix when all joints are at their midpoints as $\boldsymbol{W}_0$. Figures 3.5 and 3.6 show that both methods discussed give $\boldsymbol{W}_0 = \mathbf{I}$. This is suitable for the weighted pseudoinverse (3.39): The scaling of $\boldsymbol{W}$ does not affect the solution of the optimization problem[5], because the weighting term in the objective function is its only term, so there are no tradeoffs against other terms. Therefore $\boldsymbol{W}_0 = \beta\mathbf{I}$ gives the same analytical solution for all β, and $\boldsymbol{W}_0 = \mathbf{I}$ is a reasonable choice.

However, in the case of weighted damped least-squares, the objective function minimizes three terms: the task space velocity error; the joint velocity norm when the configuration is nearly singular; and the weighted joint velocities. To avoid unnecessary errors, we must have $\boldsymbol{W}_0 = \mathbf{0}$, which is easily achieved by modifying (3.50) to

$$w_i = \begin{cases} \left|\frac{\partial H}{\partial q_i}\right| & \text{when } \Delta\left|\frac{\partial H}{\partial q_i}\right| \geq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.54}$$

3.5 Thrust allocation

This section derives the joint position-dependent thrust configuration matrix $\boldsymbol{B}(\boldsymbol{q}) \in \mathbb{R}^{n_d \times n_t}$ and presents how to use it to solve the thrust allocation problem—the task of calculating thruster inputs as a function of the desired resultant forces on the vehicle.

Here, n_d is the number of controlled degrees of freedom and n_t is the number of

[5]Except in cases when the scaling is so small or large that it leads to numerical inaccuracies.

thrusters. The transformation matrix

$$
{}^{0}\boldsymbol{T}_{t_i} = \begin{pmatrix} {}^{0}\boldsymbol{R}_{t_i} & {}^{0}\boldsymbol{p}_{t_i} \\ \boldsymbol{0} & 1 \end{pmatrix}, \tag{3.55}
$$

defines the position and orientation of thruster t_i in the base frame, where the rotation of the thruster is defined such that a positive control input produces a force along the positive x-axis of the thruster frame, as illustrated in the simulation model in Figure 3.7.

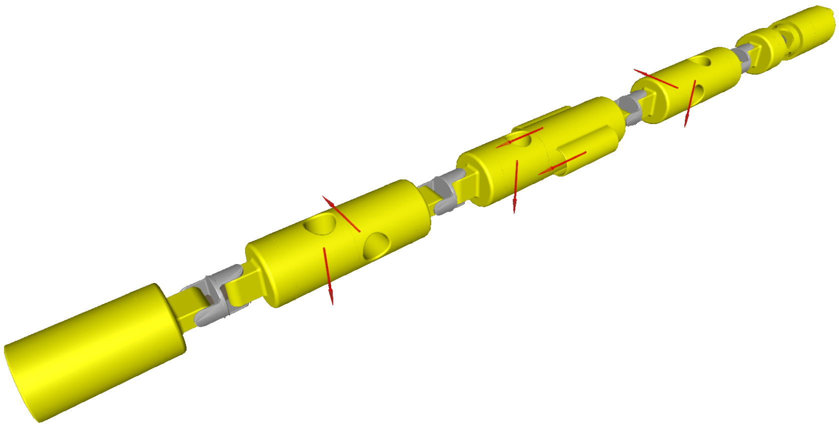


Figure 3.7: An example of thruster placement on a USM, showing the positive x-axes of the thruster frames as red arrows.

Each thruster defines to one column of

$$
\boldsymbol{B}(\boldsymbol{q}) = \begin{pmatrix} \boldsymbol{b}_1 & \boldsymbol{b}_2 & \dots & \boldsymbol{b}_{n_t} \end{pmatrix}, \tag{3.56}
$$

and each column, assuming $n_d = 6$, is calculated as [44]

$$
\boldsymbol{b}_i = \begin{pmatrix} \mathbf{I}_{3\times 3} \\ -\boldsymbol{S}({}^{0}\boldsymbol{p}_{t_i})^{\mathrm{T}} \end{pmatrix} {}^{0}\boldsymbol{R}_{t_i} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \in \mathbb{R}^6, \tag{3.57}
$$

which is simply an extraction of the first column (x-direction) of the adjoint map of the base-to-thruster transformation. For $n_d < 6$, the elements of $\boldsymbol{b}_i$ corresponding to the uncontrolled degrees of freedom can simply be removed, so that $\boldsymbol{b}_i \in \mathbb{R}^{n_d}$.

The next step is to determine ${}^0\boldsymbol{p}_{t_i}$ and ${}^0\boldsymbol{R}_{t_i}$ as a function of the joint variables. If thruster t_i is fixed to frame $\mathcal{F}_j$ with the transformation ${}^j\boldsymbol{T}_{t_i}$, then the thruster transformation from the base frame can be calculated as

$${}^0\boldsymbol{T}_{t_i}(\boldsymbol{q}) = {}^0\boldsymbol{T}_j(\boldsymbol{q})\,{}^j\boldsymbol{T}_{t_i}. \tag{3.58}$$

Repeating the process for $t_i = 1, 2, \ldots, n_t$ fully defines the thrust configuration matrix. If forces and moments expressed in the inertial frame are used, the equations above can be modified by adding the inertial to base frame transformation ${}^I\boldsymbol{T}_0$.

The thrust configuration matrix $\boldsymbol{B}$ defines a mapping from the forces produced by each thruster to the resulting forces and moments exerted on the USM. In a similar fashion to the inverse kinematics problem at the differential level, in Section 2.2, we need to find the opposite mapping: From desired forces and moments to the corresponding thruster forces.

There are several ways of solving

$$\boldsymbol{\tau} = \boldsymbol{B}(\boldsymbol{q})\boldsymbol{u}, \tag{3.59}$$

for the thruster force vector $\boldsymbol{u}$, where $\boldsymbol{\tau}$ are the forces and moments exerted by the thrusters on the vehicle. An in-depth discussion of this topic, however, is outside the scope of this thesis. Instead, the most straightforward solution is used, which simply uses the pseudoinverse [6, Ch. 12]:

$$\boldsymbol{u} = \boldsymbol{B}^{\dagger}(\boldsymbol{q})\,\boldsymbol{\tau}, \tag{3.60}$$

where

$$\boldsymbol{B}^{\dagger} = \boldsymbol{B}^{\mathrm{T}}\left(\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}\right)^{-1}. \tag{3.61}$$

Pseudoinverse thrust allocation is sufficient given that the thrust configuration matrix $\boldsymbol{B}$ is full-rank, which is true by Assumption 5.

Chapter 4

Implementation and simulation

This chapter analyzes the potential of GJM based USM control, both for vehicles subjected to and not subjected to hydrostatic restoring forces. After an introduction to the simulation environment and model, the simulation results and their discussion will be presented. Several scenarios based on similar velocity and position reference trajectories are simulated, to allow comparison between the different components and alternatives of the control scheme.

4.1 Vortex simulation model

The simulations are run on a combination of MATLAB/Simulink [53] and Vortex Studio [58]. The control system is implemented in Simulink, while the dynamic model is implemented in Vortex. Vortex performs real-time, multi-body hydrodynamic simulations with visualization. Vortex is previously used in [3], which also describes a previous iteration of the Vortex implementation of the USM model.

The USM simulation model in Vortex is based on a CAD drawing of the Eelume USM, which is a 3.3 m long vehicle weighing 81.3 kg. The real-life USM is shown in Figure 1.1f and its representation in Vortex is shown in Figure 4.1. The CAD drawing and Vortex model have been developed externally, not as part of this thesis.

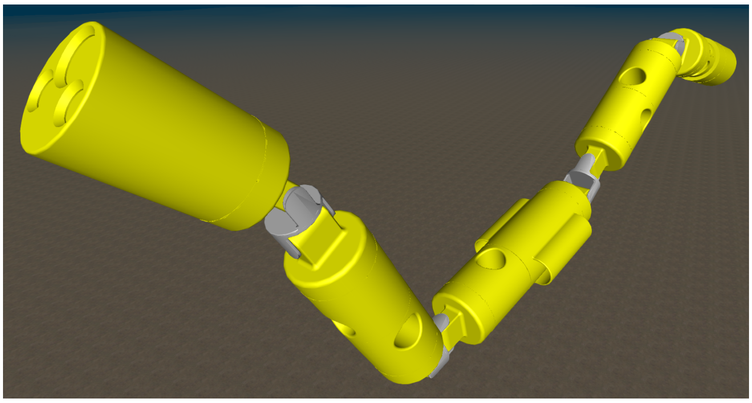


Figure 4.1: The USM simulation model, as seen in Vortex. The end-effector link is to the left, and the base link to the right.

The model consists of nine links connected by eight joints. The eight joints are arranged pairwise, in double compound joints that rotate in yaw and pitch. Five of the nine links are larger "modules" (colored yellow, lengths ranging from 37 to 73 cm), and the remaining four are smaller "couplings" between yaw and pitch joints (colored gray, 10 cm long). The three central modules are equipped with a total of seven thrusters, enabling 6 DOF actuation in most joint configurations. The motorized joints are modeled as first order systems, with a step response time constant $\tau_q = 100$ ms, shown in Figure 4.2.

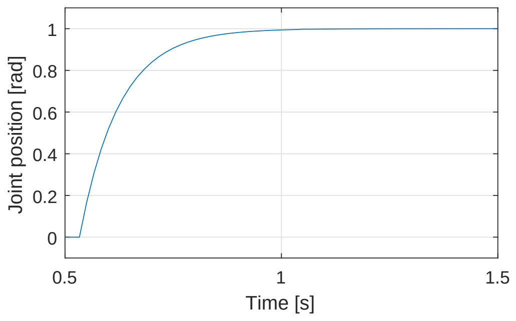


Figure 4.2: Unit step response of the joint motors in the Vortex USM model. The time constant is $\tau_q = 100$ ms.

4.2 Vortex-Simulink interface

In Appendix B, Figure B.1 shows the top-level of the implementation of the control system in MATLAB/Simulink. Figure B.2 shows the interface in Simulink that wraps the Vortex-Simulink connection block. With this custom interface, both applications can be linked to each other and run in parallel. The Simulink connections to the block inputs are sent to the Vortex model, while the Simulink connections to the block outputs are retrieved from the Vortex model, in real time, as Figure 4.3 shows. As is visible in Figure B.1, the inputs to the simulator are joint position commands $q_{1,\mathrm{cmd}}, \ldots, q_{8,\mathrm{cmd}}$, in radians, and thruster commands $u_{1,\mathrm{rpm}}, \ldots, u_{7,\mathrm{rpm}}$, in revolutions per minute. The Vortex outputs are the measured joint positions $q_{1,\mathrm{m}}, \ldots, q_{8,\mathrm{m}}$; transformations from the global Vortex frame to the front module ${}^{\mathrm{vx}}\boldsymbol{T}_{\mathrm{front}}$ and to the tether module (base link) ${}^{\mathrm{vx}}\boldsymbol{T}_{\mathrm{tether}}$; as well as the linear and angular velocities of the front and tether modules.

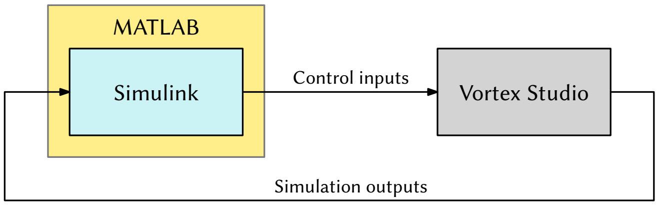


Figure 4.3: Connections between MATLAB/Simulink and Vortex.

The signals are vectorized and processed in a wrapper subsystem to simplify the interface between Vortex and Simulink. The wrapper defines initial conditions so that the transformation outputs are guaranteed to be valid at all times.[1] All signals are processed to assure that algebraic loops do not occur. The transformations outputs are also further transformed to allow for arbitrary definitions of the base and of the end-effector frames, as well as of the inertial frame. Angular and linear velocity outputs are transformed correspondingly.

[1]Vortex outputs zero matrices for all transformations at the first time step.

4.3 Control design model

For specification of the GJM and other parts of the control scheme, the kinematic model of Chapter 2 is used. As this is a general kinematic model for robotic manipulators, it can describe the kinematics of a USM without modification. In addition to the kinematic parameters, the masses, inertia matrices, COM and COB positions, and displaced volumes are specified for each link of the model. This provides the properties required for GJM control, as well as the properties governing the hydrostatic forces. It is not necessary to define hydrodynamic parameters for the control model. Appendix A contains the full specification of the control design model parameters used in the simulations in this chapter.

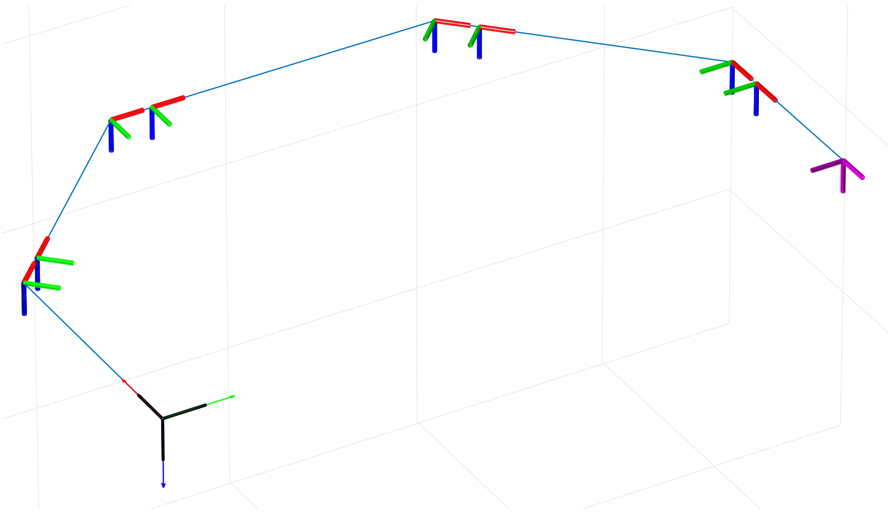


Figure 4.4: The geometry of the simulation model. The black coordinate frame is the base frame, the pink frame is the end-effector frame, and the remaining frames are the joint frames.

4.4 Overview of simulations

The simulations will be seen in the context of three example use cases:

1. Human-controlled teleoperation for end-effector positioning: The human pilot

can compensate for errors in the trajectory tracking. If the end-effector moves slower than expected, they can simply move the joystick further. With a human in the loop [44, Ch. 5], the accuracy requirements are not strict—it is only required that the output velocities be approximately proportional to the inputs. It is also important that a zero input guarantees a near-zero output

2. Autonomous positioning of sensor equipment, such as a camera: This situation does not need very precise positioning. For a camera, it would be reasonable to require a maximum position error of 10 cm. End-effector orientation accuracy is only important for non-gimballed cameras.

3. Autonomous valve-turning, using a grabber tool attached to the end-effector. Valve-turning is an intervention task that requires high precision. Although it depends on the valve and grabber tool, a reasonable limit is an accuracy on the centimeter-level, for instance, no more than 2 cm position error. See the valves and tool used in [59].

To investigate which components of the control system are applicable in which situations, nine simulation cases will be run. The first five cases are run on the Vortex model without hydrostatic forces, while the last four are run on the full model with hydrostatics included. Case 0 acts as a reference point for those that follow by simulating traditional control using base stationkeeping and fixed-base inverse kinematics for the manipulator control. Cases 1–3 and 5–7 simulate the the system in open loop, with velocity feedback, and with position feedback, on the models without and with hydrostatics, respectively. Case 4 investigates the accuracy difference of a quick motion versus a slow motion, and Case 8 is a repetition of Case 7 with the thruster-based rotation damping functionality of Section 3.2.7 activated.

The damped least-squares inverse kinematics solution (Section 3.3.2) is used in all simulations. The control law presented in Section 3.2.5 is used for velocity feedback, and the law presented in Section 3.2.6 for position feedback.

All cases simulate 3 DOF linear motion control of the end-effector in inertial space, which is applicable to many tasks and use cases, such as positioning a gimballed camera, and light intervention with a multi-DOF end-effector tool. Other tasks may require control of fewer or more DOF, such as valve turning with a simple grabber (5 DOF), or camera direction control (2 DOF). Any combination of the 6 spatial DOFs are possible to control when using the GJM, by extracting the appropriate sub-matrix. For 3 DOF velocity control, the desired values are given as linear velocities

$$ {}^{I}\boldsymbol{v}_{E,d} \equiv \begin{pmatrix} u_d & v_d & w_d \end{pmatrix}^{\mathrm{T}} \tag{4.1} $$

while for position control the desired values are x, y, and z positions

$$ {}^{I}\boldsymbol{p}_{E,d} \equiv \begin{pmatrix} x_d & v_d & w_d \end{pmatrix}^{\mathrm{T}}, \tag{4.2} $$

where subscript "d" denotes the desired value, as in Section 3.2.5. The velocity and position variables adhere to standard notation by The Society of Naval Architects and Marine Engineers [60].

The reference velocity and position trajectories that will be given are similar or identical to each other for all simulation scenarios, to allow comparison of performance. In general, the motion commanded corresponds to pulling the end-effector back, left, and up, starting from a nearly outstretched configuration.

Due to Vortex software technicalities, all simulations begin at $t = 0$ with the USM in its home position. This is however not a good choice, as it is a kinematically singular configuration. Therefore, the USM is commanded to "twitch" into the non-singular configuration

$$ \boldsymbol{q}(t = 0) = \frac{\pi}{180^\circ} \begin{pmatrix} 0^\circ & 5^\circ & 0^\circ & -5^\circ & 0^\circ & 5^\circ & 0^\circ & -5^\circ \end{pmatrix}^{\mathrm{T}} \tag{4.3} $$

immediately after the start of each simulation, as shown in Figure 4.5. This configuration is chosen in order to move the joints sufficiently far away from the singularity while generating minimal net forces on the vehicle. The twitch causes some initial velocities, visible at the beginning of the simulations. The simulations trajectories begin only after a full second has passed, to allow the velocities to die out.

4.5 Simulations without restoring forces

The simulations in this section are run with zero restoring forces and moments, which is achieved by defining the COM of each link to align with its COB.

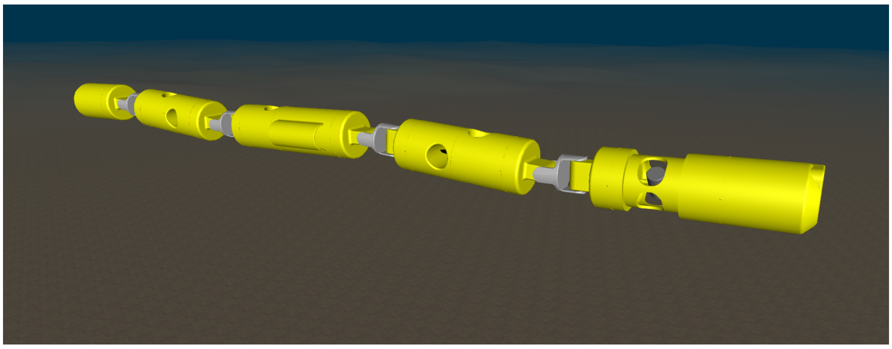


Figure 4.5: The initial USM configuration, used for all simulations unless otherwise noted.

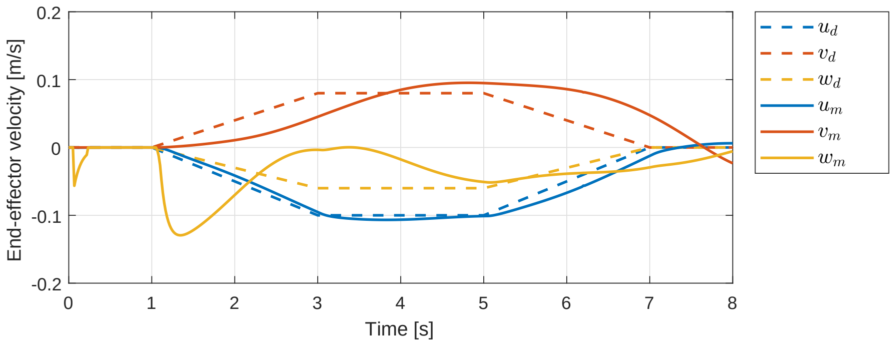
4.5.1 Case 0: Traditional control (non-GJM)

This section will present a simulation of the “traditional” control alternative to the GJM-based control schemes presented later, to provide a basis for comparison.

The traditional approach for floating base manipulation is to hold the base stationary enough to use inverse kinematic control with the fixed-base Jacobian, in other words, to control the manipulator the same way as an industrial manipulator. The GJM is not involved. For this to work, it is necessary to have a stationkeeping algorithm of sufficient performance, and actuators to fulfill its control command. The base must be controlled in 6 DOF, and any deviation from perfect setpoint regulation will normally appear as errors in the end-effector control.

For the simulation, the controller presented in [52] and discussed in Section 3.2.7 is used, in its original, full pose control form. The position gain is set to $200 \cdot \mathbf{I}$, the orientation gain to $400 \cdot \mathbf{I}$, and the derivative gain to $50 \cdot \mathbf{I}$. The end-effector velocity and position trajectories are shown in Figure 4.6, and snapshots of the configuration throughout the maneuver are given in Figure 4.8.

The velocity tracking is very inaccurate, but the position trajectory is better, as the velocity errors change signs throughout the simulation. Better performance will be possible to achieve with a better stationkeeping algorithm, for instance by defining a feed-forward term based on the commanded joint trajectory. Another improvement would be to use the singularity-robust task-priority approach of [3]. The most relevant information is however in Figure 4.7. It displays the commands for each thruster, which



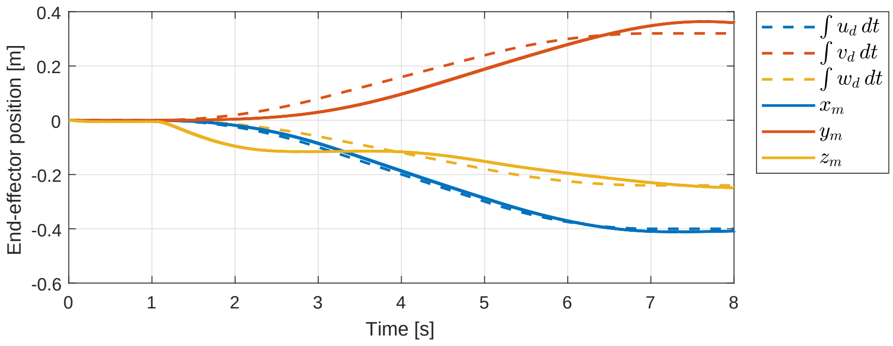


Figure 4.6: (Case 0) End-effector velocity (above) and position accuracy (below) for traditional control (stationkeeping with fixed-base inverse kinematics). Dashed lines are desired values, solid lines are measurements.

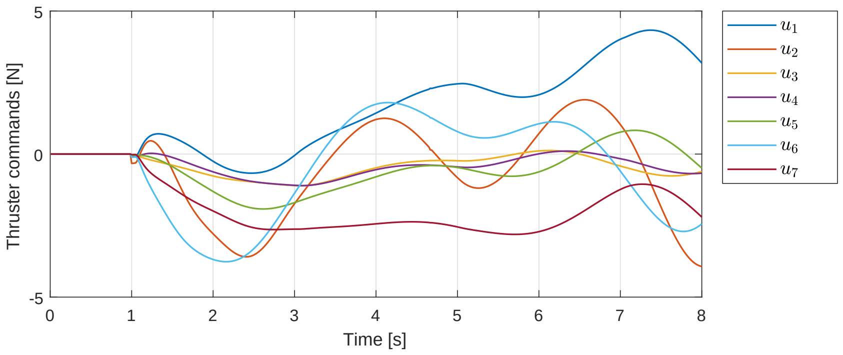
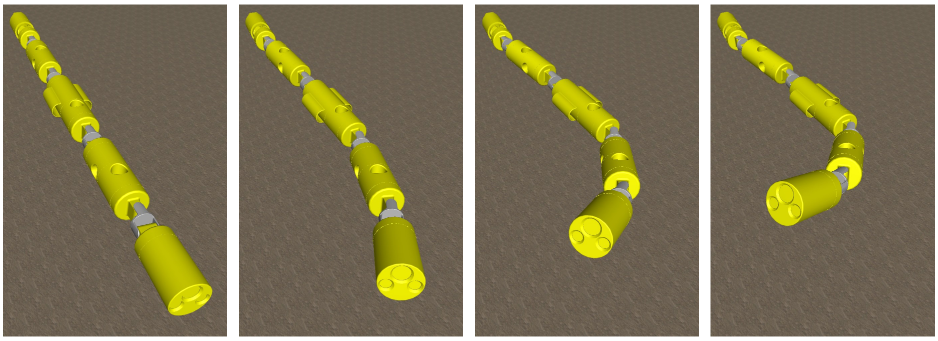


Figure 4.7: (Case 0) Commanded force from each thruster, in newtons, for traditional control (stationkeeping with fixed-base inverse kinematics).



(a) $t \approx 1$ s (b) $t \approx 3$ s (c) $t \approx 5$ s (d) $t \approx 7$ s

Figure 4.8: (Case 0) Snapshots of the configuration change for traditional control (stationkeeping with fixed-base inverse kinematics), at 2 s increments of the simulation time t. Notice how the base stays fixed while the end-effector moves.

range roughly from negative five to five newton. Thrusters are usually more expensive to actuate than motorized joints. With stationkeeping and fixed-base inverse kinematics, it is necessary to actuate both, leading to a significant increase in energy expenditure, which is especially problematic for battery-powered vehicles.

4.5.2 Case 1: Open-loop control

This simulation scenario expresses to what extent the GJM is applicable for USM control, despite the unmodeled hydrodynamic effects. Figure 4.9 compares the desired and the measured end-effector velocities, and the integrals of the desired velocities to the measured positions. USM configurations during the operation are shown in Figure 4.10.

The end-effector velocities follow the trajectory with only small deviations, no larger than approximately 0.02 cm/s. The errors are mostly found in the sway and heave velocities during the steady-state segment of the trajectory. This is presumably due to the unmodeled hydrodynamic effects—the tendency of the vehicle to “swim” when moving the joints. A small part of the transient error can also be attributed to the joint motor dynamics (Figure 4.2). (Research on USR propulsion, such as [10], exploit the swimming effect for locomotion.) That the surge velocity is relatively accurate is likely because negative surge velocity corresponds to a contraction of the whole USM. When contracting under GJM control, the head and tail are both pulled inward, which

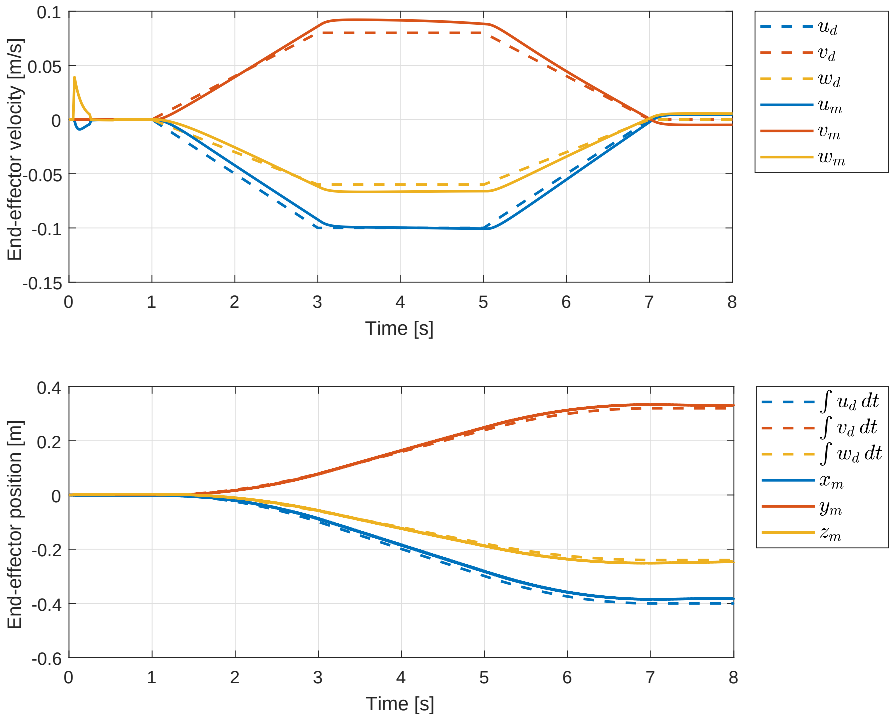


Figure 4.9: (Case 1) End-effector velocity (above) and position accuracy (below) for open-loop GJM control. Dashed lines are desired values, solid lines are measurements.

will create reaction forces that approximately cancel out. On the other hand, reaction forces due to sway and heave motion of the head are not compensated for by movement with canceling hydrodynamic forces. Despite this, the norm of the position deviation at $t = 8$ s is no more than about 2.2 cm, making this suitable for all three use cases listed in Section 4.4. However, the position error must be expected to accumulate over time, so the applicability depends on the required time horizons for autonomous tasks.

4.5.3 Case 2: Velocity feedback

With velocity feedback, the velocity tracking errors are reduced compared to the open-loop errors in Case 1, from 2.2 cm (Figure 4.9) to 1.5 cm (Figure 4.11). It is task-dependent whether the difference is significant or not, based on the accuracy needed. This case is

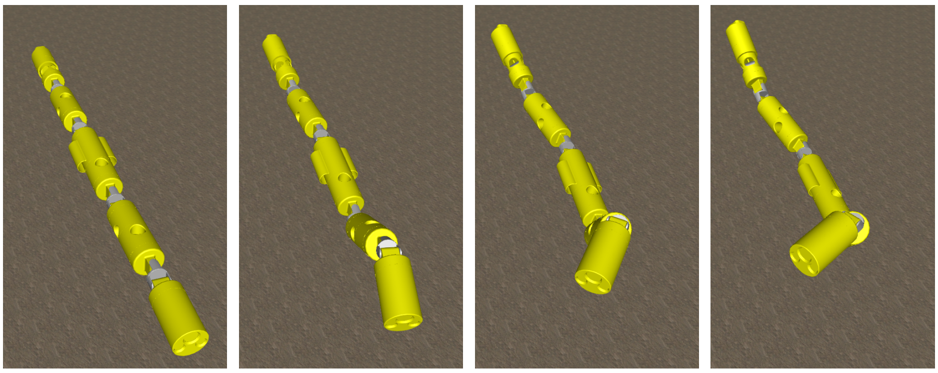

(a) $t \approx 1$ s (b) $t \approx 3$ s (c) $t \approx 5$ s (d) $t \approx 7$ s

Figure 4.10: (Case 1) Snapshots of the configuration change for open-loop GJM control, at 2 s increments in the simulation time t. Notice how all parts of the robot move to allow correct end-effector motion.

applicable to the same use cases as Case 1. Velocity feedback is useful and should be used whenever velocity estimates are available, but the difference is not crucial. However, a more sophisticated velocity controller will be able to increase the advantage of velocity feedback.

4.5.4 Case 3: Position feedback

This simulation uses CLIK with $\boldsymbol{K}_{\text{clik}} = 3 \cdot \mathbf{I}_{3\times3}$, discussed in Section 3.2.6. As shown in Figure 4.12, the velocities differ from their references in order to follow the position trajectory closely. The position references are generated by integrating the velocity references over time,

$$x_d(t) \equiv \int_0^t u_d(t)\,\mathrm{d}t, \tag{4.4}$$
$$y_d(t) \equiv \int_0^t v_d(t)\,\mathrm{d}t, \tag{4.5}$$
$$z_d(t) \equiv \int_0^t w_d(t)\,\mathrm{d}t. \tag{4.6}$$

The norm of the position error, at the end of the simulation, is on the order of millimeters, and the performance overall is excellent (Figure 4.12). With position feedback, the control scheme is sufficiently accurate for all three use cases.

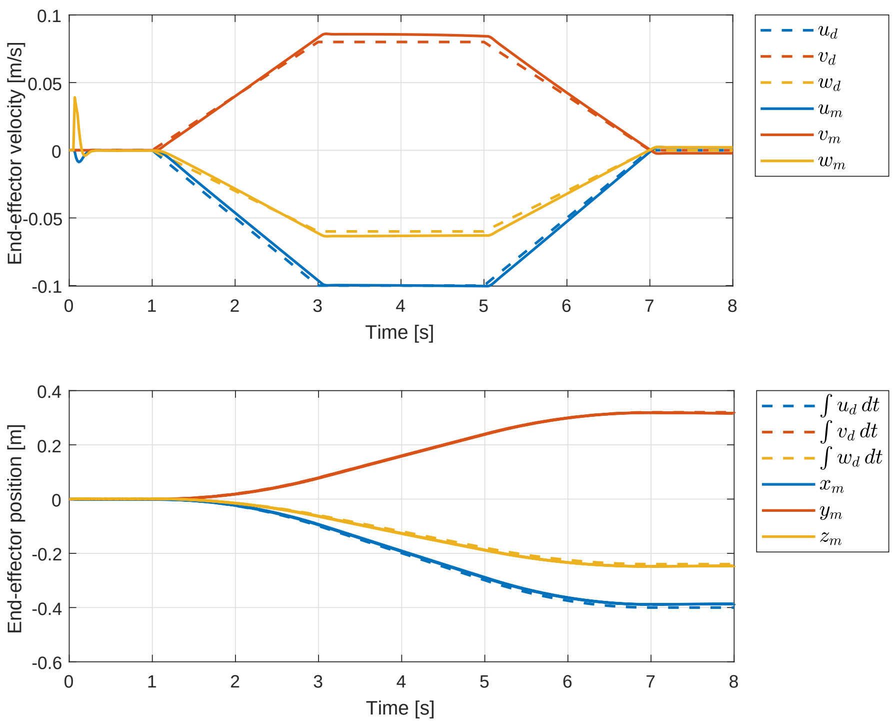


Figure 4.11: (Case 2) End-effector velocity (above) and position accuracy (below) for closed-loop GJM control with velocity feedback. Dashed lines are desired values, solid lines are measurements.

4.5.5 Case 4: Accuracy for slow vs. fast motion

One hypothesis of this thesis is that the hydrodynamic effects weaken significantly as the velocity decreases (Section 1.2). A velocity trajectory modified from that of Figure 4.9 is repeated in Figure 4.13, in one faster and one slower variant.

The largest velocity error of the slow maneuver is 1.2 cm/s (in the y-direction), while the largest error for the fast maneuver is 7.1 cm/s (Figure 4.13). It does indeed appear to be true that the velocity tracking errors increase and decrease with velocity, but not so much that they become negligible even at velocities on the order of 1 cm/s, which is the maximum velocity command of the slow maneuver given.

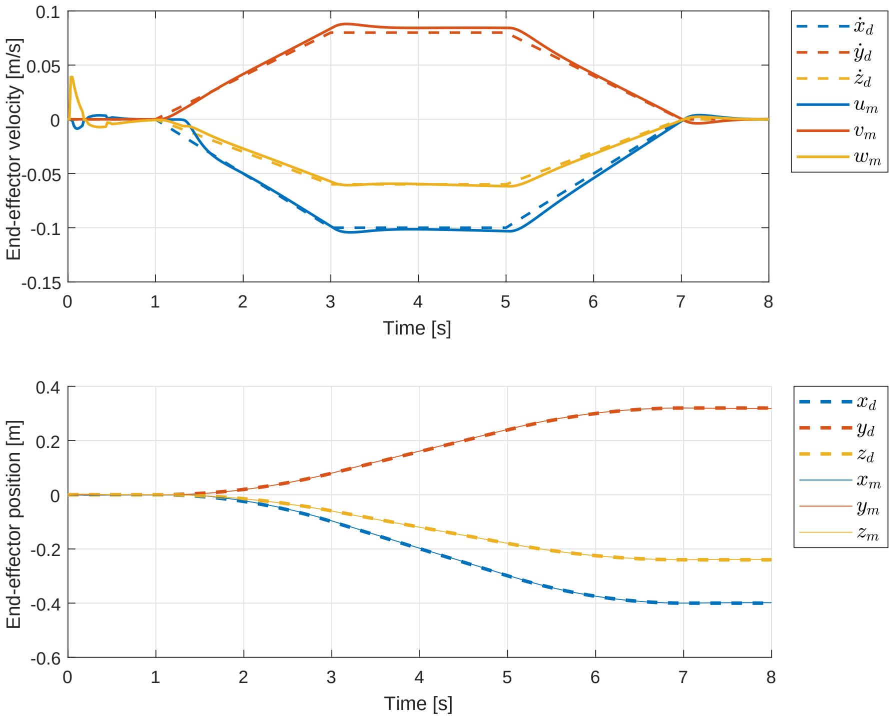


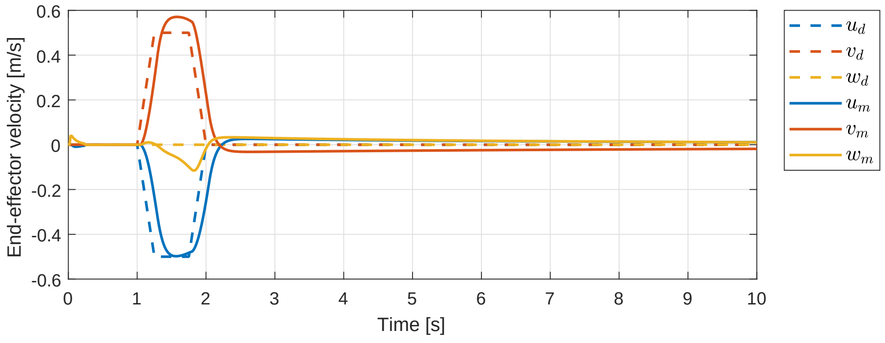
Figure 4.12: (Case 3) End-effector velocity (above) and position accuracy (below) for closed-loop GJM control with position feedback. Dashed lines are desired values, solid lines are measurements.

4.6 Simulations with restoring forces

The simulations below are run on the full model, with nonzero restoring moments. They give an idea of the magnitude of the restoring force disturbance, and how well velocity and position feedback can counteract it. Simulation horizons are doubled in the remaining cases, to show the oscillations caused by hydrostatics.

4.6.1 Case 5: Open loop

For comparison, the same open-loop maneuver as that in Case 1 (Section 4.5.2) is repeated with the full model. Unsurprisingly, substantial errors ensue, as no steps are taken to avoid the disturbances. This is evident from Figure 4.14. The velocities



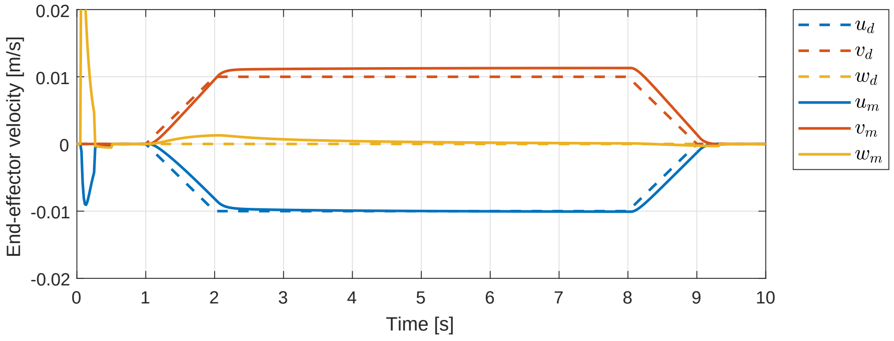


Figure 4.13: (Case 4) End-effector velocity accuracy for fast (above) and slow (below) velocities using open-loop GJM control. Dashed lines are desired values, solid lines are measurements.

follow their references to some extent, but poorly, especially in the heave direction. The increased error is guaranteed to be caused by hydrostatic forces, as the model, control scheme, and parameters are otherwise identical to those in Figure 4.9. In addition to the poor following of the trajectory during the first half of the simulation, sinusoidal oscillations appear during the latter half (Figure 4.14). These oscillations are due to the rotation induced by the restoring moments, and they oscillate around the hydrostatic equilibrium point. Pure open-loop control is insufficient for all three sample use cases when hydrostatic forces are present. The root mean square (RMS) of the position error norm during the simulation is 26.3 cm, which is almost half of the 56.6 cm total desired position change.

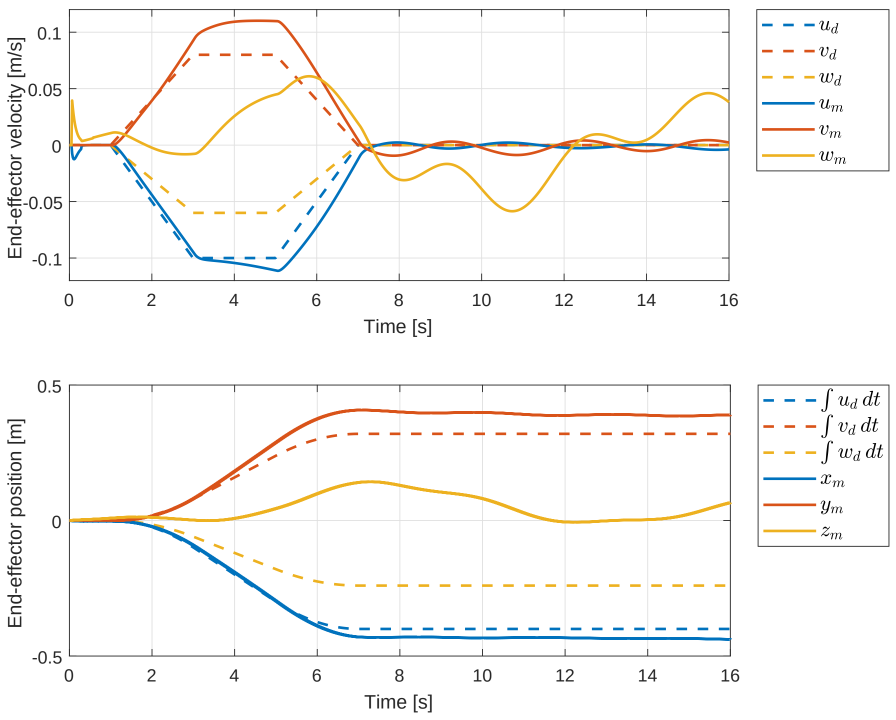


Figure 4.14: (Case 5) End-effector velocity (above) and position accuracy (below) for open-loop GJM control, subjected to hydrostatic forces. Dashed lines are desired values, solid lines are measurements.

4.6.2 Case 6: Velocity feedback

Similar to in Case 2 (Section 4.5.3), velocity feedback with the restoring forces model (Figure 4.15) reduces the errors to roughly half of their original magnitude (Figure 4.14). The velocity tracking is as such significantly better than without feedback, but not good enough to be considered acceptable. Although the x and y-directions are precise, the heave direction is not. Inaccurate z motion is as problematic as inaccurate x and y motion, but easier to solve as it can be measured and compensated for at the position level by a simple pressure sensor. Measurements of x and y position require much more complex and expensive equipment. Section 4.6.3 shows the result of repeating the simulation with full 3 DOF position feedback, and it is reasonable to assume that a similar performance for the heave direction could be achieved with heave feedback only.

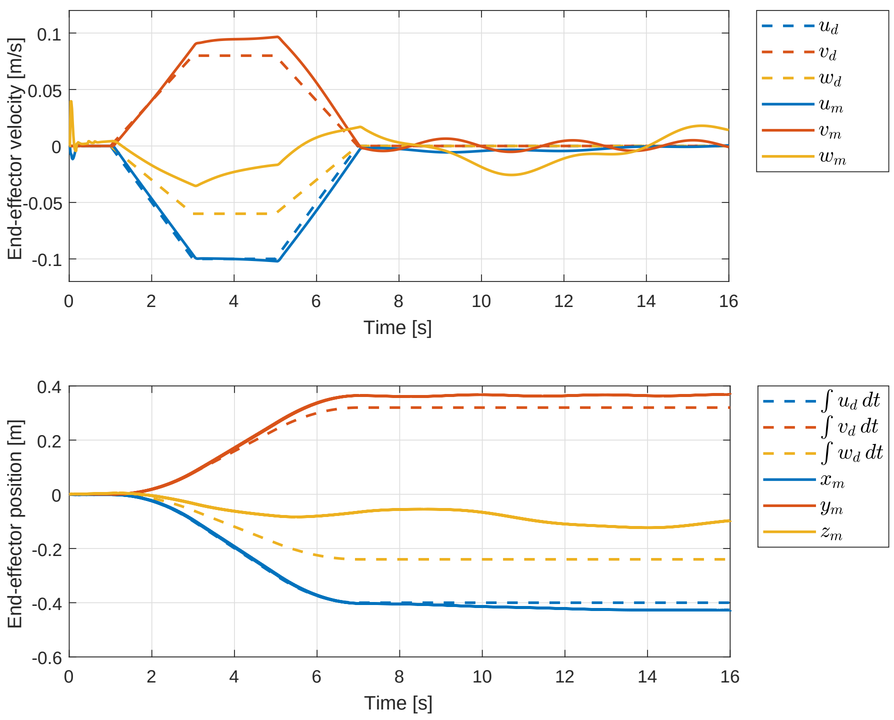


Figure 4.15: (Case 6) End-effector velocity (above) and position accuracy (below) for closed-loop GJM control with velocity feedback, subjected to hydrostatic forces. Dashed lines are desired values, solid lines are measurements.

If the USM is teleoperated, this still is somewhat usable, as the end-effector mostly moves in the right direction, and the pilot would be able to compensate for the poor accuracy by increasing the corresponding velocity commands. However, for both of the autonomous use-cases, the performance here is too poor. A better velocity feedback algorithm might alleviate the problem, but, given sufficient sensor data, position feedback is a safer solution, as shown in the next section. The RMS of the position error norm is 13.3 cm, which mirrors the apparent halving of the error, which was 26.3 cm previously, without feedback (Section 4.6.1).

4.6.3 Case 7: Position feedback

With position feedback, and the same CLIK algorithm as in Section 4.5.4, the tracking performance is greatly improved from the previous case, and the RMS of the position error norm is reduced from 13.3 cm (Figure 4.15) to only 1.1 cm (Figure 4.16), fulfilling the requirements of all three use-cases. There are some small transient errors in the position tracking, and some oscillatory errors after the reference trajectory ends. Compare to Figure 4.12, which shows an identical simulation run on the model without restoring forces. Although the oscillations will eventually die out on their own, waiting for the slow decay is not desirable, as it reduces accuracy, and wastes power by moving the joints to keep the end-effector stable. The transient tracking is however excellent (Figure 4.16) and can be improved further by a better-tailored position controller.

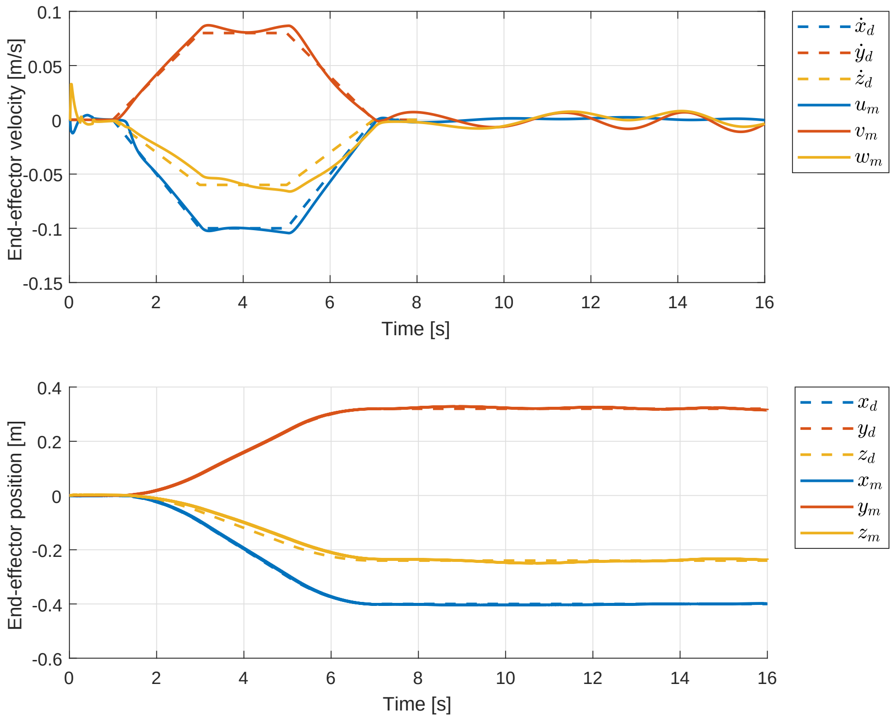


Figure 4.16: (Case 7) End-effector velocity (above) and position accuracy (below) for closed-loop GJM control with position feedback, subjected to hydrostatic forces. Dashed lines are desired values, solid lines are measurements.

4.6.4 Case 8: Position feedback and thruster damping

Section 3.2.7 describes how the attitude of the vehicle can be actively controlled toward its hydrostatic equilibrium point, in order to damp the oscillations that are present in Sections 4.6.1 (Case 5), 4.6.2 (Case 6), and 4.6.3 (Case 7). The attitude controller is activated immediately after the trajectory ends, at $t = 7$ s. The controller succeeds in damping out the oscillations and does so swiftly. With the oscillations damped, the RMS of the position error norm has been reduced to as little as 0.4 cm.

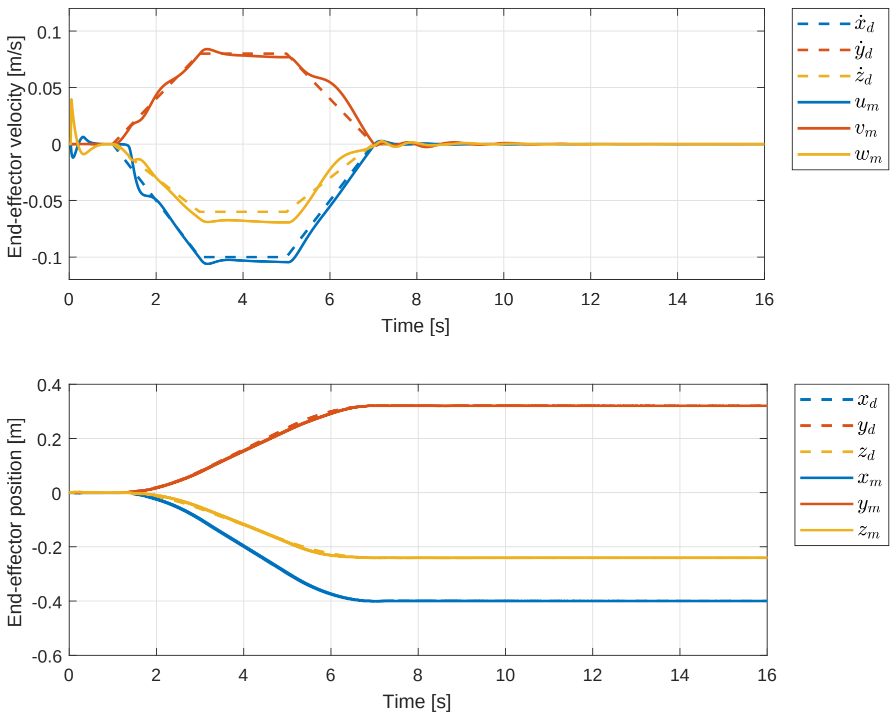


Figure 4.17: (Case 8) End-effector velocity (above) and position accuracy (below) for closed-loop GJM control with position feedback and thruster stopping, subjected to hydrostatic forces. Dashed lines are desired values, solid lines are measurements.

Whether thruster damping is preferable depends on the cost of actuating the thrusters. Typically, thruster actuators spend more energy than joint motors, and this is the case for the Eelume USM in Figure 1.1f. The Eelume USM's thrusters consume up to 350 W, and its joint motors up to 70 W. The change in total power consumption should be weighed against the importance of the accuracy improvement thruster stopping gives, which will depend on the use case.

Chapter 5

Conclusion

The thesis has demonstrated that GJM-based control of underwater manipulators, specifically USMs, is possible. Despite the GJM being originally formulated based on the assumption of no external forces, it is shown to give good performance also for underwater systems that are subject to hydrostatic and hydrodynamic forces. Hydrodynamic simulations have verified the control scheme, and shown which variants are applicable in which situations. In particular, open-loop control was found to be sufficient for teleoperation, autonomous sensor positioning, and autonomous intervention, when the vehicle is freed from hydrostatic disturbance, assuming short task durations. When hydrostatic forces are present or task durations are long, position feedback is necessary to reach the same level of performance.

The accuracy improves with access to velocity and position measurements. Accuracy is further improved with a thruster-based method implemented to damp oscillations that occur after an end-effector trajectory, due to restoring forces. Figure 5.1 shows how adding velocity and position feedback reduces the RMS end-effector position error, and how the thruster damping can reduce it further, for nonzero restoring forces. While the accuracy is good, a main drawback of GJM control is that the size of the reachable workspace shrinks due to the low base inertia, but this is a limitation of the physical structure, not of the control system.

In summary, the system achieves centimeter-level accuracy even in the face of hydrostatic restoring forces, given that reliable position estimates are available for

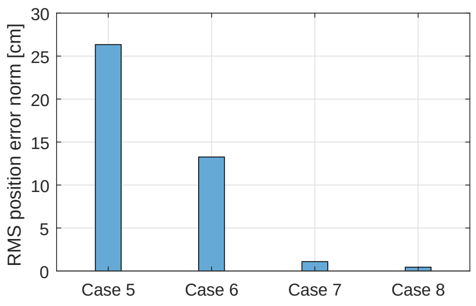


Figure 5.1: Comparison of root mean square of the position error norm for variations of the control system, using the restoring forces simulation model. Case 5 is open-loop; Case 6 with velocity feedback; Case 7 with position feedback; and Case 8 with position feedback and thruster damping.

feedback control. The accuracy makes it a viable and more efficient alternative to traditional control methods, for both teleoperation and autonomous work.

5.1 Prospects for future research

As the USM is a recently developed robot architecture, their control systems are in an early phase of development. This opens for many possible directions for future work, some of which are discussed below.

Already mentioned is the possibility of improving the velocity and position controllers. The controllers used for simulations in this thesis are very rudimentary, and proportional-integral-derivative (PID) control or more advanced, possibly model based alternatives deserve investigation.

As discussed in Section 1.2, the control scheme here does not consider hydrodynamic forces. It was observed, especially in Figure 4.9, that the effects of hydrodynamics on the USM are small but existent. Estimation of the hydrodynamics and subsequent incorporation into the control system should give improved accuracy and better predictability of the performance. Research by McLain and colleagues, especially [35], can act as a starting point.

Control of any redundant manipulator requires redundancy resolution. The proposed control system performs redundancy resolution with a focus on singularity avoidance. Singularity avoidance is necessary, as discussed in Section 3.1.2, but possible to combine with additional objectives. Some examples are:

- Use the reaction null-space to perform reactionless manipulation, to minimize the disturbance of the base link. Reactionless manipulation is particularly useful for vehicles that have cameras or other sensor equipment attached to the base link, such as the Eelume USM which has a base swivel camera. Whether the RNS is large enough to be realistically useful requires examination.

- Define a "hydrostatic null space", which is the space inside of which the hydrostatic restoring forces always are zero. It is straightforward to prove that such a workspace can exist: If the COM and COB of each link is vertically aligned and the vehicle is horizontal, any yaw motion will gives pure translation of the COM and COB, not creating restoring forces. Depending on the size of the workspace, it might be possible to do useful work within it, thereby avoiding reliance on restoring force compensation.

- Alternatively, define avoidance of hydrostatic forces as a secondary task. This is a better solution than a hydrostatic null space if the null space is too small to keep operation entirely inside it.

- Avoid configurations where the thruster configuration is ill-conditioned. With the 7-thruster USM used for simulations in this thesis, the thrust configuration matrix is usually able to maintain full rank, giving the ability to actuate all 6 DOF, but some configurations cause rank deficiency. Avoiding these configurations will assure that proper actuation from the thrusters is possible at all times.

Another useful addition is to use the thrusters to extend the manipulator workspace. Normally, the thrusters are dormant during manipulation, but if the manipulator approaches the boundary of its workspace, the thrusters can move the whole vehicle closer to the point of interest. This can be implemented as an on/off or a gradually activating effect. A workspace must be defined for this method, and a good choice seems to be the path-independent workspace (PIW), or *guaranteed workspace*, which is a workspace within which it is guaranteed that a free-floating manipulator can reach any point, regardless of its path toward that point [56]. Alternatively, a dexterity measure, such as that proposed in [61], or one that is based on the SVD [62], can determine when to switch to vehicle repositioning. These methods would need a trajectory tracking algorithm to reposition the COM of the manipulator, the motion of which is decoupled from the manipulation itself. Therefore, it is straightforward to implement on top of the manipulation algorithm presented here.

Appendix A

Control design model

This appendix contains tables that define the properties of the USM model used in the control design.

Table A.1: Geometric properties of each link of the USM model. Defines the location the link's center of mass and center of buoyancy, the location of the origin of the next link, and the link's displaced volume. For the model with zero restoring forces, ${}^{i}z_{cm,i} = 0$ for all i.

Link	Next link pos. [mm]			COM pos. [mm]			COB pos. [mm]			Volume [dm^3]
i	${}^{i}x_{i+1}$	${}^{i}y_{i+1}$	${}^{i}z_{i+1}$	${}^{i}x_{cm,i}$	${}^{i}y_{cm,i}$	${}^{i}z_{cm,i}$	${}^{i}x_{cb,i}$	${}^{i}y_{cb,i}$	${}^{i}z_{cb,i}$	∇_i
0	620	0	0	303	0	0	303	0	0	143
1	104	0	0	52	0	0	52	0	0	60
2	584	0	0	292	0	70	292	0	0	127
3	104	0	0	52	0	0	52	0	0	60
4	726	0	0	362	0	70	362	0	0	98
5	104	0	0	52	0	0	52	0	0	60
6	584	0	0	292	0	70	292	0	0	127
7	104	0	0	52	0	0	52	0	0	60
8	370	0	0	193	0	0	193	0	0	78

Table A.2: Mass and inertia properties of the USM model. The inertia properties are given as the six unique elements of the standard inertia tensor $\boldsymbol{I}_i \in \mathbb{R}^{3\times 3}$.

Link	Mass [kg]	Inertia tensor [kgm^2]					
i	m_i	$I_{xx,i}$	$I_{yy,i}$	$I_{zz,i}$	$I_{xy,i}$	$I_{xz,i}$	$I_{yz,i}$
0	14.3	0.0579	0.3737	0.3737	0.0000	0.0000	0.0000
1	6.0	0.0163	0.0103	0.0163	$1.647 \cdot 10^{-4}$	$-9.143 \cdot 10^{-9}$	$3.372 \cdot 10^{-7}$
2	12.7	0.0487	0.3479	0.3479	$-9.456 \cdot 10^{-5}$	$-9.513 \cdot 10^{-5}$	$-1.017 \cdot 10^{-6}$
3	6.0	0.0163	0.0103	0.0163	$1.647 \cdot 10^{-4}$	$-9.143 \cdot 10^{-9}$	$3.372 \cdot 10^{-7}$
4	9.8	0.0506	0.3671	0.3826	$-4.580 \cdot 10^{-5}$	$-1.238 \cdot 10^{-5}$	$2.825 \cdot 10^{-6}$
5	6.0	0.0163	0.0103	0.0163	$1.647 \cdot 10^{-4}$	$-9.143 \cdot 10^{-9}$	$3.372 \cdot 10^{-7}$
6	12.7	0.0487	0.3480	0.3480	$-9.564 \cdot 10^{-5}$	$-9.436 \cdot 10^{-5}$	$2.5433 \cdot 10^{-6}$
7	6.0	0.0163	0.0103	0.0163	$1.647 \cdot 10^{-4}$	$-9.143 \cdot 10^{-9}$	$3.372 \cdot 10^{-7}$
8	7.8	0.0302	0.0966	0.0968	$3.4143 \cdot 10^{-5}$	$9.1293 \cdot 10^{-6}$	$1.3929 \cdot 10^{-6}$

Table A.3: Positions and orientations of the thrusters relative to their parent frame. The parent of a thruster is the link to which it is attached. Position is given in the parent link frame. θ is the azimuth angle and ϕ is the elevation angle.

Thruster	Parent link	Thruster pos. [mm]			Thruster rot. [rad]	
t_i	i	${}^i x_{t_i}$	${}^i x_{t_i}$	${}^i x_{t_i}$	θ_{t_i}	ϕ_{t_i}
t_1	2	237	0	0	0	$-\pi/2$
t_2	2	347	0	0	$\pi/2$	0
t_3	4	278	-100	0	0	0
t_4	4	278	100	0	0	0
t_5	4	488	0	0	0	$-\pi/2$
t_6	6	237	0	0	$\pi/2$	0
t_7	6	347	0	0	0	$-\pi/2$

Appendix B

Implementation details

This appendix contains key illustrations of the implementation of the control system.

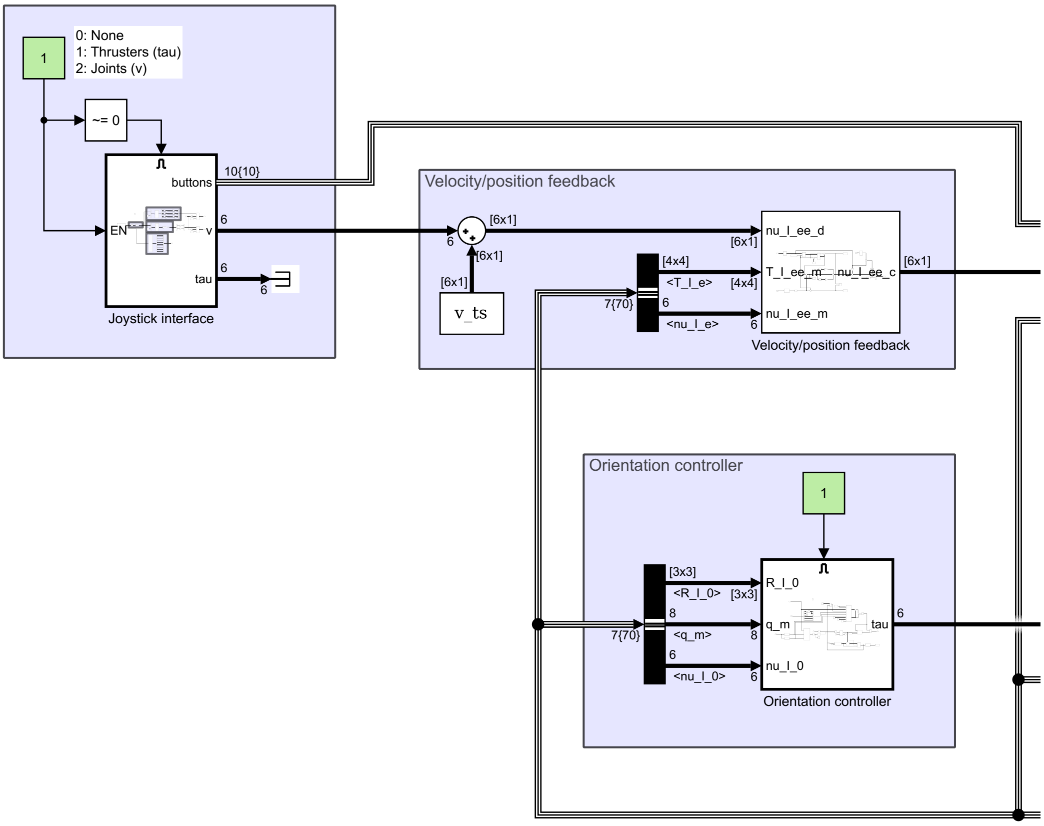


Figure B.1: Top-level view of the control system implemented in Simulink.

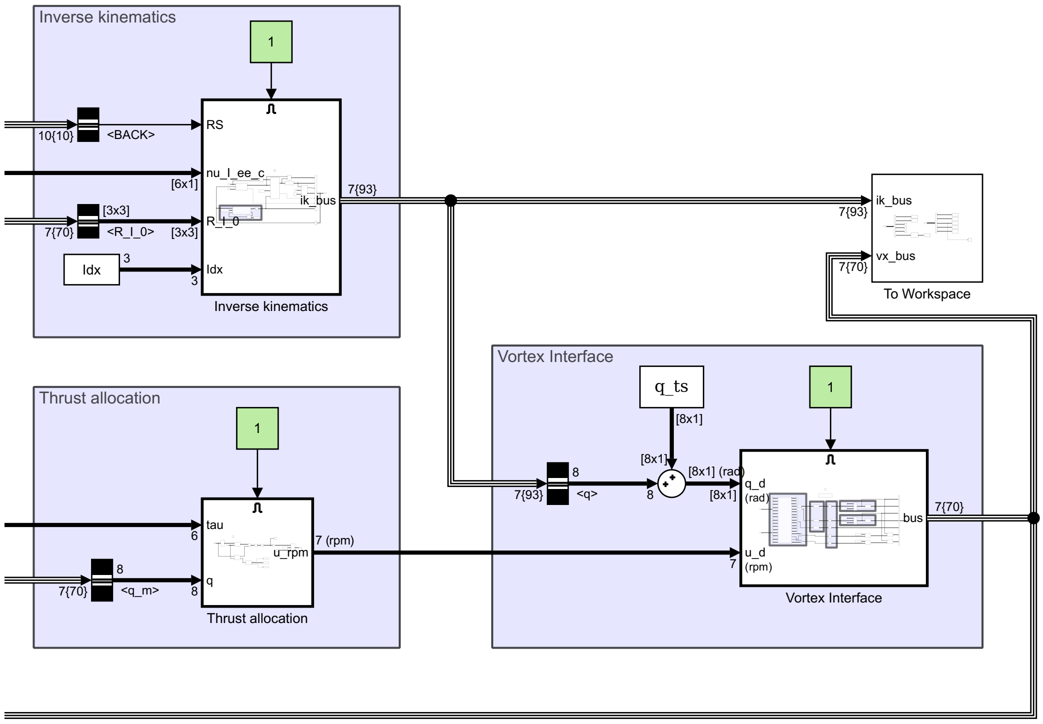


Figure B.1: Continued from last page.

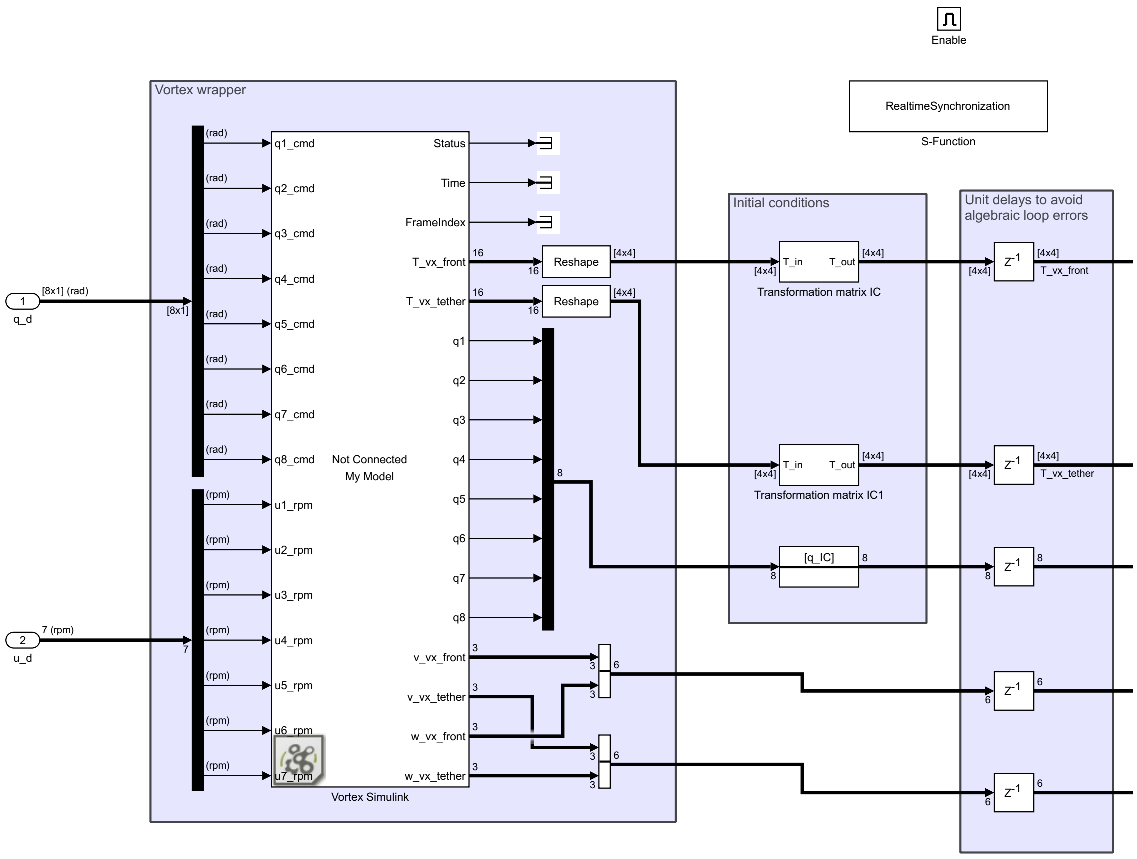


Figure B.2: The Simulink-Vortex interface.

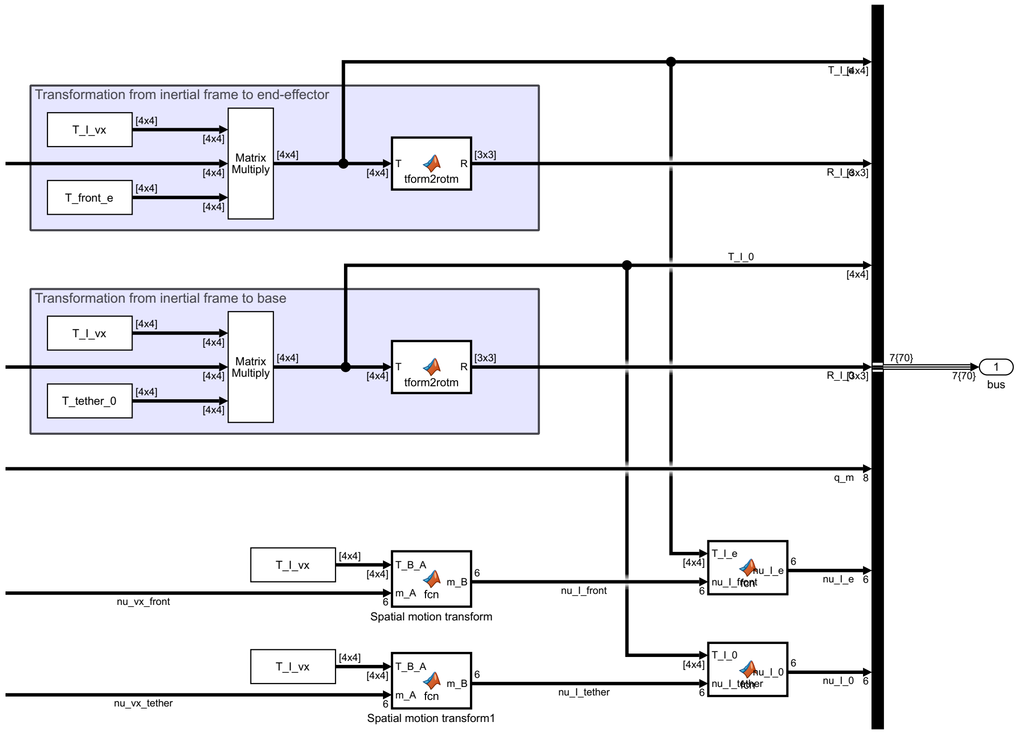


Figure B.2: Continued from last page.

Appendix C

Conference paper

This appendix provides a draft for a conference paper based on the core findings of this thesis, aimed for submission to the 2018 European Control Conference. The draft is in accordance with all formal formatting requirements of the PaperCept Conference Management System for initial submission as a contributed paper to ECC'18.

Inverse Kinematic Control of a Free-Floating Underwater Manipulator Using the Generalized Jacobian Matrix

Morten F. Amundsen, Jørgen Sverdrup-Thygeson, Eleni Kelasidi, Kristin Y. Pettersen
Norwegian University of Science and Technology (NTNU)
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics
Trondheim, Norway
Email: morten.f.amundsen@ntnu.no

***Abstract*—Traditional control of robotic manipulators has assumed that the base of the manipulator is fixed, which holds true for industrial manipulators. Underwater vehicle-manipulator systems are not fixed to their environment, but can be held fairly still by thruster actuation and a good stationkeeping algorithm, which allows traditional fixed-base manipulator control. This paper will describe a method for underwater manipulation that does not require stationkeeping, and by extension does not require thrusters. The method is based on inverse kinematics using the generalized Jacobian matrix, a generalization of the traditional fixed-base manipulator Jacobion, originally used for control of robotic manipulators in outer space. With the generalized Jacobian matrix, a free-floating manipulator can be controlled in inertial coordinates, and the base remains passive and unactuated, only moving to aid in the motion of the end-effector. The control system is verified through hydrodynamic simulations using the Vortex Studio software, and open loop control is compared to position feedback control. The simulations show that underwater manipulation without base actuation is possible, and that the accuracy can be improved with access to position estimates.**

Fig. 1. The Eelume USM, an example of an underwater snake robot. It is approximately $3.3\,\mathrm{m}$ long and weighs $80\,\mathrm{kg}$.

I. Introduction

Underwater robots exist in numerous forms and have a multitude of applications. Examples are seafloor mapping and geological sampling in research and science; construction, inspection, maintenance and repair of subsea installations in the oil and gas industry; and search and disposal of mines for the military [1].

Some tasks, especially complex intervention tasks that have yet to be automated, and also some observation tasks, are suited for remotely operated vehicles (ROVs), which are tethered to the surface and usually teleoperated [2]. Other tasks, such as wide area surveying and mapping, are usually done by autonomous underwater vehicles (AUVs), as they can operate for extended periods of time without human involvement [1].

The underwater swimming manipulator (USM) is a new class of underwater robots that combines a bio-inspired snake-like appearance with thruster actuators [3]. It is currently only produced by *Eelume* and represents a recent field of robotics research. The Eelume USM is shown in Figure 1. USMs are a continuation of the underwater snake robot (USR), which is a snake-like underwater robot designed to "swim" like an eel or a sea snake [4].

This paper will investigate how to achieve accurate and flexible underwater manipulation without thruster usage. Previous underwater vehicle-manipulator system (UVMS) and USM control has used thrusters and joint motors for manipulation [5], which is energy-inefficient due to the high power consumption of the thrusters. In contrast to the existing methods, this paper will present a solution that does not rely on thruster usage, ultimately saving power and extending the possible mission duration. The control system will be general for any UVMSs. Underwater robots float freely in the water, in contrast to an industrial robotic manipulator which is fixed to the ground. Because of this, the USM responds differently to three types of forces in particular:

1) **Reaction forces:** Moving the joints of the manipulator will induce reaction forces that disturb the position and orientation of the manipulator base [6]. The disturbance occurs because the base is floating freely in the water.

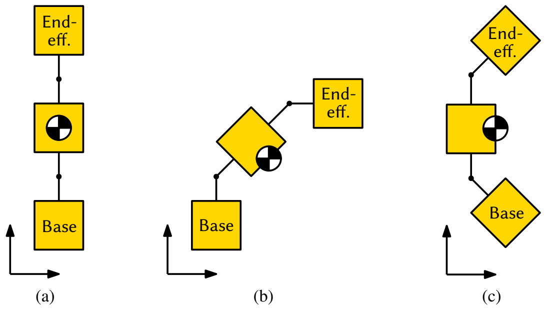


Fig. 2. Differences between fixed-base and floating-base manipulation: (a) shows the initial configuration, and (b) and (c) shows final configurations after identical joint motion. In (b), the base is fixed, and this causes the total COM to move. In (c), the base is floating, and reaction forces on the base cause it to move, while the location of the COM remains the same.

By contrast, the base of an industrial manipulator is firmly fixed to its environment and is not disturbed by joint motion. Figure 2 compares the effects of joint motion and reaction forces on fixed-base and floating-base manipulators.

2) **Hydrostatic forces:** Gravity "pulls" the center of mass (COM) down, and buoyancy "pushes" the center of buoyancy (COB) up. The COM and COB of an underwater manipulator are not necessarily aligned, and this causes rotational hydrostatic restoring forces [7, Ch. 4]. Because the COM and COB locations depend on the joint configuration, the resulting restoring forces also depend on the joint configuration.
3) **Hydrodynamic forces:** The manipulator moves through water, and is subject to hydrodynamic effects such as drag forces and increased inertia from added mass [8, Ch. 10].

The effects of the forces listed above are present for all underwater manipulators, but not to the same extent. Robots that have a large "body" with a smaller manipulator attached to it will not be strongly affected, because of the large inertia of the base. Intervention-ROVs and -AUVs are typical examples. USRs and USMs are more strongly affected. These robots have a base link that is only a small fraction of the total vehicle's inertia and size, and as a result, both reaction forces and hydrostatic forces that occur while moving the manipulator will affect them much more than ROVs and AUVs. It is therefore vital to compensate for these effects when attempting manipulation tasks.

II. Inverse Kinematic Control

A. The generalized Jacobian matrix

The generalized Jacobian matrix (GJM), introduced first in [9] and [6] provides a map between the joint velocities $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ to the velocity twist $\boldsymbol{\nu} = (\boldsymbol{v}^{\mathrm{T}}, \boldsymbol{\omega}^{\mathrm{T}})^{\mathrm{T}} \in \mathbb{R}^6$ of the end-effector in the inertial frame,

$$\boldsymbol{\nu} = \hat{\boldsymbol{J}}(\boldsymbol{q}, \boldsymbol{R})\, \dot{\boldsymbol{q}}, \tag{1}$$

where n is the number of joints, $\hat{\boldsymbol{J}} \in \mathbb{R}^{6\times n}$ is the GJM, $\boldsymbol{q} \in \mathbb{R}^n$ are the joint positions, and $\boldsymbol{R} \in SO(3)$ is the rotation from the inertial frame to the base link. Equation (1) contrasts the map of fixed-base manipulator Jacobians, which map to the end-effector velocity twist, (usually) expressed in the base link or end-effector frame. The GJM is able to map to inertial velocities by taking into account the displacement of the end-effector that occurs due to conservation of momentum.

A full derivation of the GJM is available in [10]. The kinematic parameters of the vehicle, its link masses and inertia matrices, the joint configuration, and the orientation of the base frame, are necessary to define the GJM. The constant parameters are typically available for a given robot. The joint configuration can be measured for instance with rotary encoders, and the base orientation can be estimated based on measurements from an inertial measurement unit (IMU).

The GJM was originally developed for manipulators mounted on free-floating vehicles in the near-vacuum of space, which experience negligible external forces [11]. Conservation of momentum applies for these systems. As described in Section I, underwater vehicles are subject hydrodynamic effects, and the assumption of no external forces does not hold for these systems. However, the aim here is to investigate whether the hydrodynamic effects are small enough to make GJM-based control feasible.

B. Singularities of the generalized Jacobian matrix

A manipulator is said to be in a singular configuration when its Jacobian is singular, or rank deficient. When the Jacobian is singular, some of its task-space velocities are impossible to reach, and its inverse mapping is not well-defined. For square singular Jacobians, the inverse matrix is undefined. In the neighborhood of a singularity, the task velocities may be theoretically reachable, but only with excessive joint velocities. Large joint velocities can cause spurious and unpredictable motion, and may exceed the physical ability of the joint actuators.

Nearly all manipulators have *kinematic singularities*, which are singularities due to its kinematic structure. They can occur at the workspace boundary, and at points where joint axes align. The GJM also exhibits *dynamic singularities*, which are dependent on the dynamic properties of the vehicle. Kinematic and dynamic singularities reflect physical limitations and are only avoidable by staying away from them, not by, for instance, expressing the problem differently.

USMs and USRs are subject to more significant reaction forces than typical free-floating space manipulators. For a normal snake robot, the base link is of approximately the same size and mass as the other links, while for a satellite manipulator (a typical application of GJM control) the base is usually significantly heavier than the entire manipulator. An example is the "Engineering Test Satellite VII", which is a 140 kg arm mounted on a 2550 kg base, used for in-orbit GJM experiments [11]. The workspace of a free-floating snake robot is therefore fairly small in comparison to its kinematic dimensions. Singularities at workspace boundary and inside the workspace are rarely far away, and it is necessary to avoid them in a way that minimizes the resulting tracking errors.

C. Inverse kinematics

Equation (1) maps from joint velocities to end-effector velocities. For control, it is necessary to define the inverse mapping—finding the joint velocities needed to achieve a desired end-effector velocity. However, $\hat{\boldsymbol{J}}$ is not generally invertible. Methods that solve this are termed inverse kinematic methods. This section will discuss an alternative for inverse kinematic control of USMs. The inverse of (1) can be written

$$\dot{\boldsymbol{q}} = \boldsymbol{f}(\hat{\boldsymbol{J}})\boldsymbol{\nu}. \tag{2}$$

Damped least-squares inverse kinematics [12] is a solution of (2) that avoids singular configurations by damping the joint velocities in the neighborhood of the singularities. It is the minimizer of

$$\left\|\boldsymbol{\nu} - \hat{\boldsymbol{J}}\dot{\boldsymbol{q}}\right\|^2 + \lambda^2\|\dot{\boldsymbol{q}}\|^2 \tag{3}$$

with respect to joint velocities. The solution is

$$\dot{\boldsymbol{q}} = \left(\hat{\boldsymbol{J}}^{\mathrm{T}}\boldsymbol{J} + \lambda^2\mathbf{I}\right)^{-1}\hat{\boldsymbol{J}}^{\mathrm{T}}\boldsymbol{\nu}, \tag{4}$$

or equivalently [13],

$$\dot{\boldsymbol{q}} = \hat{\boldsymbol{J}}^{\mathrm{T}}\left(\hat{\boldsymbol{J}}\hat{\boldsymbol{J}}^{\mathrm{T}} + \lambda^2\mathbf{I}\right)^{-1}\boldsymbol{\nu}. \tag{5}$$

Equation (4) is preferable as it requires fewer operations to compute, assuming $n > 6$ [13].

The damping term $\lambda^2\|\dot{\boldsymbol{q}}\|^2$ in (3) is designed to slow down joint motion when the manipulator is near a singular configuration, which can be achieved by choosing λ^2 according to [14]

$$\lambda^2 = \begin{cases} 0 & \text{when } \sigma_{\min} \geq \epsilon \\ \left[1 - \left(\frac{\sigma_{\min}}{\epsilon}\right)^2\right]\lambda^2_{\max} & \text{otherwise,} \end{cases} \tag{6}$$

where $\sigma_{\min}$ is the smallest singular value (which is a measure of the proximity to a singularity [15]), ϵ is the singular value threshold below which the damping becomes active, and $\lambda_{\max}$ is the maximum damping factor. With (6), the solution is a pure task velocity error minimization when the manipulator is sufficiently far from any singularities. As it approaches a singular configuration, the joint velocity damping term will dominate.

D. Position feedback

Accurate estimates of end-effector position may be hard to obtain. The position is possible to estimate by dead-reckoning, which combines and integrates velocity and acceleration level measurements, but the estimate will drift. If only acceleration measurements are available, the drift error will be quadratic in time (due to the double integration), and with velocity measurements, it will be linear in time (due to the single integration) [16]. Estimators based on double integration are usually not accurate for a long enough time to be useful for precision tasks. For this reason, Doppler velocity log (DVL) measurements are normally used in inertial navigation systems (INSs) [17]. These indirect estimation methods are not perfect, but are often the only available methods for ocean operations away from human-made structures.

Various methods exist for direct position measurement underwater. While global navigation satellite systems (GNSSs) are unavailable underwater, a similar method based on triangulation with ranges to acoustic transponders, called long baseline (LBL), is possible. Another possibility is ultra-short baseline (USBL), using a sonar array for position measurements [17]. The drawback of all underwater position measurement methods is that the sensor equipment must be installed on site—sensors on board the vehicle are not sufficient for most methods. An exception is camera-based simultaneous localization and mapping (SLAM), which is a viable method in situations with good visibility and does not require equipment installed in the environment [18].

A simple method for incorporating position data into the manipulator control scheme is to use closed-loop inverse kinematics (CLIK) [19]. CLIK was developed to remove the accumulated integration error from joint trajectory reconstruction [13], but is here used to also account for position error that occurs during underwater GJM-based manipulation. A first-order CLIK algorithm, adapted from [13] to give position but not orientation feedback, is

$$\dot{\boldsymbol{q}} = \hat{\boldsymbol{J}}^{\dagger}\begin{pmatrix}\boldsymbol{v} + \boldsymbol{K}(\boldsymbol{p} - \hat{\boldsymbol{p}}) \\ \boldsymbol{\omega}\end{pmatrix} + \left(\mathbf{I} - \hat{\boldsymbol{J}}^{\dagger}\hat{\boldsymbol{J}}\right)\dot{\boldsymbol{q}}_0, \tag{7}$$

where $\boldsymbol{K}$ is a constant, positive definite gain matrix, $\boldsymbol{p}$ is the end-effector position in the inertial frame according to the reference trajectory, and $\hat{\boldsymbol{p}}$ is a measurement or estimate of $\boldsymbol{p}$. The last term allows exploitation of the redundancy, with an arbitrary joint velocity vector $\dot{\boldsymbol{q}}_0$. We remove the inverse kinematics of the CLIK, so that (7) reduces to

$$\boldsymbol{\nu}_c = \begin{pmatrix}\boldsymbol{v} + \boldsymbol{K}(\boldsymbol{p} - \hat{\boldsymbol{p}}) \\ \boldsymbol{\omega}\end{pmatrix} \tag{8}$$

where $\boldsymbol{\nu}_c$ is the velocity twist command extracted from (7). The advantage is that (8) can be used with any inverse kinematic method. Note again that this is a very simple position feedback control scheme. It is likely possible to achieve better performance with other methods, but the aim here is to illustrate the potential benefits of closed-loop control with position feedback compared to open-loop control—not to investigate the feedback algorithms thoroughly. A diagram of the closed loop system, combining position feedback (8) with damped least-square inverse kinematics (4), is displayed in Figure 3.

III. Simulations

This section analyzes the potential of GJM based control. Two simulation scenarios will be compared: Open-loop inverse kinematics, and position-feedback closed-loop inverse kinematics.

Both situations simulate 3 degree of freedom (DOF) linear motion control of the end-effector in inertial space, which is applicable to many tasks and use cases, such as positioning

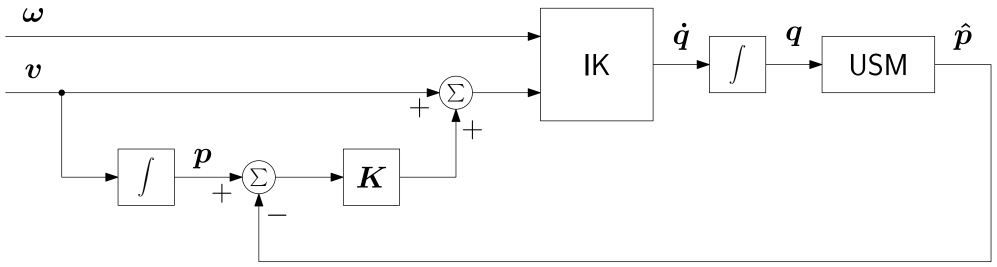


Fig. 3. Diagram of the closed-loop inverse kinematics control system, using position feedback.

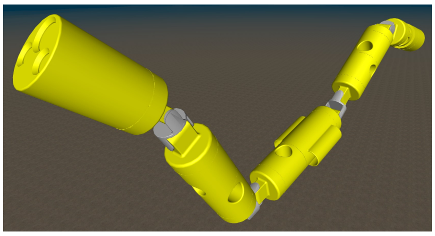


Fig. 4. The USM simulation model, as seen in Vortex

a gimballed camera, and light intervention with a multi-DOF end-effector tool. Any combination of the 6 spatial DOFs are possible to control when using the GJM, by extracting the appropriate sub-matrix. The reference trajectories are given as linear velocities

$$\boldsymbol{v} \equiv (u_d, v_d, w_d)^{\mathrm{T}}, \tag{9}$$

and as linear positions

$$\boldsymbol{p} \equiv (x_d, v_d, w_d)^{\mathrm{T}}. \tag{10}$$

The reference velocity and position trajectories are identical in both simulation scenarios, to allow comparison of performance. The trajectory corresponds to pulling the end-effector back, left, and up, starting from a nearly outstretched configuration.

A. *Simulation environment*

The simulations are run on a combination of MATLAB/Simulink [20] and Vortex Studio [21]. The control system is implemented in Simulink, while the dynamic model is implemented in Vortex. Vortex performs real-time hydrodynamic and visual simulations, based on a link-by-link model of the USM, where each part has its kinematic and dynamic properties specified. The Vortex model is a continuation of the model presented in [3]. The main differences are that the model used here has links of varying lengths, and the thrusters have been moved and increased in number from five to seven. The newer model used here is shown in Figure 4.

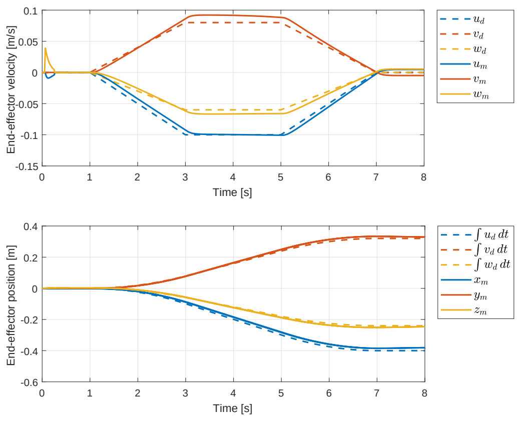


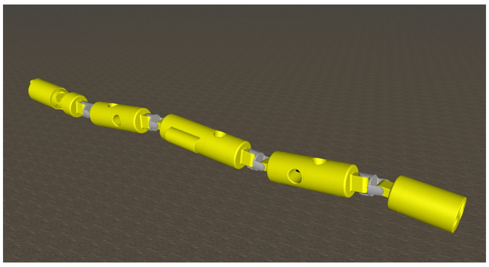
Fig. 5. End-effector velocity (above) and position trajectories (below) for the open-loop maneuver. Dashed lines represent the reference trajectory, and solid lines the simulation output. The lower sub-figure compares the integrated velocity trajectory to the measured end-effector positions.

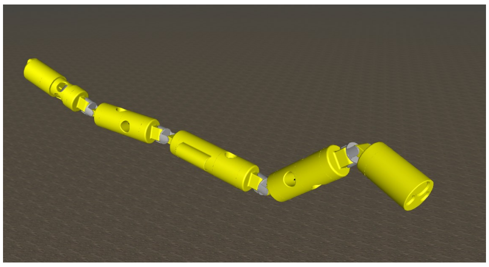
B. *Open loop inverse kinematics*

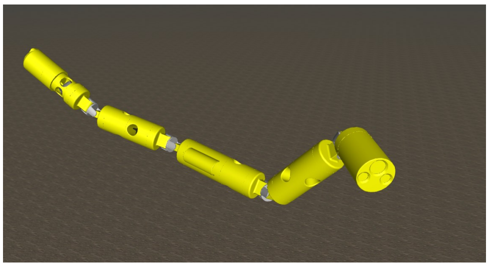
The desired and measured end-effector velocity are compared in Figure 5, together with a comparison of the measured position versus the integrals of the velocity references. The chosen trajectory corresponds to moving the end-effector backward, to the right, and up (as seen from the aft of the vehicle). Vehicle configurations during the operation are shown in Figure 6.

This simulation scenario expresses to what extent the GJM is applicable for underwater manipulator control, despite the unmodeled hydrodynamic effects. Figure 5 compares the desired and the measured end-effector velocities, and the integrals of the desired velocities to the measured positions. USM configurations during the operation are shown in Figure 6.

The end-effector velocities follow the trajectory with only small deviations, no larger than approximately $0.02\,\mathrm{cm/s}$. The errors are mostly found in the sway and heave velocities during the steady-state segment of the trajectory. This is presumably due to the unmodeled hydrodynamic effects—the tendency of the vehicle to "swim" when moving the joints. A small part of the transient error can also be attributed


(a) $t \approx 1$


(b) $t \approx 3$


(c) $t \approx 5$

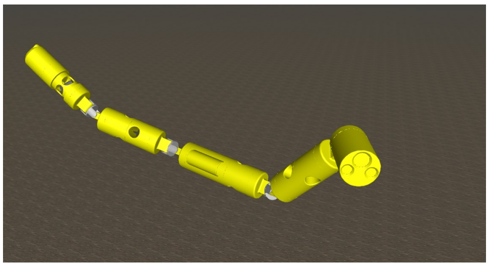

(d) $t \approx 7$

Fig. 6. Snapshots of the vehicle configurations throughout the open-loop maneuver, at even intervals of the simulation time t.

to the joint motor dynamics, which are modeled as a first order system with a $100\,\mathrm{ms}$ time constant. That the surge velocity is relatively accurate is likely because negative surge velocity corresponds to a contraction of the whole manipulator. When contracting under GJM control, the head and tail are both pulled inward, which will create reaction forces that approximately cancel out. On the other hand, reaction

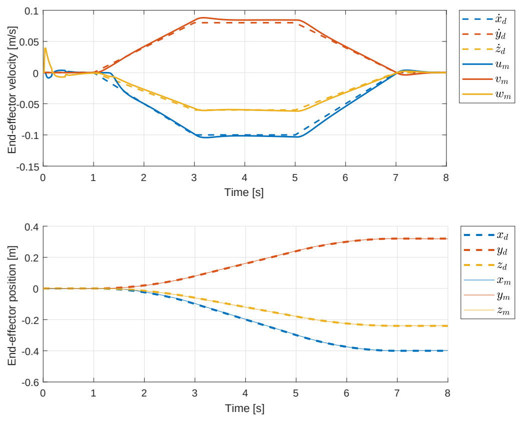


Fig. 7. End-effector velocity (above) and position trajectories (below) for the closed-loop maneuver. Dashed lines represent the reference trajectory, and solid lines the simulation output. The upper sub-figure compares the derivative of the position trajectory to the measured end-effector velocity.

forces due to sway and heave motion of the head are not compensated for by movement with canceling hydrodynamic forces. Despite this, the norm of the position deviation at $t = 8\,\mathrm{s}$ is no more than about $2.2\,\mathrm{cm}$, making this suitable for teleoperation and autonomous operation. However, the position error must be expected to accumulate over time, so the applicability depends on the required time horizons for autonomous tasks.

C. *Closed loop inverse kinematics*

This simulation uses CLIK with $\boldsymbol{K} = 3 \cdot \mathbf{I}_{3\times3}$. As shown in Figure 7, the velocities differ from their references in order to follow the position trajectory closely. The position references are generated by integrating the velocity references over time,

$$x_d(t) \equiv \int_0^t u_d(t)\,\mathrm{d}t, \tag{11}$$

$$y_d(t) \equiv \int_0^t v_d(t)\,\mathrm{d}t, \tag{12}$$

$$z_d(t) \equiv \int_0^t w_d(t)\,\mathrm{d}t. \tag{13}$$

The norm of the position error, at the end of the simulation, is on the order of millimeters, and the performance overall is excellent (Figure 7). With position feedback, the control scheme is sufficiently accurate for teleoperation and autonomy, without restrictions on the time horizon.

IV. Conclusion

A control system for free-floating underwater manipulators has been proposed. The system has been shown through hydrodynamic simulations to perform well, given that sufficient measurement data is available. The most significant drawback is that the size of the reachable workspace shrinks a lot due to the low base inertia, but this is a limitation of the physical structure, not of the control system.

The specific application dictates the level of precision needed for underwater manipulation. If the vehicle is piloted by a human, using a joystick or other device to give end-effector velocity commands, main concern is that the resulting end-effector velocities must be somewhat proportional to the velocity references given. Errors in magnitude can fairly easily be countered by the pilot: If they notice that the end-effector moves too fast or slow, they can simply adjusting how far they push the joystick. A human pilot will also typically have access to a live camera feed, which allows the pilot to perform manual position feedback, by monitoring the position of the end-effector on the video. The *human-in-the-loop* situation is elaborated on in [22].

If the system navigates with full autonomy, the requirements on control accuracy are much stricter, as no human is present to detect and correct unexpected errors. For intervention tasks, for instance valve turning, the end-effector position accuracy needs to be on the order of centimeters (compare for instance the valves used in [23]).

Inverse kinematics with the GJM without feedback is sufficient for teleoperation. Expanding the control system with position feedback increases accuracy and therefore eases the operator's work, but is not essential. For autonomous operation, position feedback is clearly preferable, as it accounts for the errors that occur due to the unmodeled hydrodynamic effects, and allows the desired centimeter precision.

In summary, the system achieves centimeter-level precision given that good position estimates are available for feedback control, making both teleoperation and autonomous operation possible.

References

[1] J. Yuh, "Design and control of autonomous underwater robots: A survey," *Autonomous Robots*, vol. 8, no. 1, pp. 7–24, 2000.

[2] J. Yuh, G. Marani, and D. R. Blidberg, "Applications of marine robotic vehicles," *Intelligent Service Robotics*, vol. 4, no. 4, pp. 221–231, oct 2011.

[3] J. Sverdrup-Thygeson, E. Kelasidi, K. Y. Pettersen, and J. T. Gravdahl, "The Underwater Swimming Manipulator – A Bio-Inspired AUV," in *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*. Tokyo, Japan: IEEE, 2016, pp. 387–395.

[4] E. Kelasidi, P. Liljebäck, K. Pettersen, and J. Gravdahl, "Biologically Inspired Swimming Snake Robots: Modeling, Control and Experimental Investigation," *IEEE Robotics and Automation Magazine*, 2015.

[5] J. Sverdrup-Thygeson, E. Kelasidi, K. Y. Pettersen, and J. T. Gravdahl, "A control framework for biologically inspired underwater swimming manipulators equipped with thrusters," in *Proc. 10th IFAC Conference on Control Applications in Marine Systems (CAMS)*, vol. 49, no. 23, Trondheim, Norway, 2016, pp. 89–96.

[6] Y. Umetani and K. Yoshida, "Resolved Motion Rate Control of Space Manipulators with Generalized Jacobian Matrix," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 303–314, jun 1989.

[7] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley, apr 2011.

[8] P. J. From, J. T. Gravdahl, and K. Y. Pettersen, *Vehicle-Manipulator Systems*, 1st ed., ser. Advances in Industrial Control. London: Springer, 2014.

[9] Y. Umetani and K. Yoshida, "Continuous path control of space manipulators mounted on OMV," *Acta Astronautica*, vol. 15, no. 12, pp. 981–986, dec 1987.

[10] K. Yoshida and Y. Umetani, "Control of Space Manipulators with Generalized Jacobian Matrix," in *Space robotics: Dynamics and control*, Y. Xu and T. Kanade, Eds. Springer, 1993, ch. 7, pp. 165–204.

[11] K. Yoshida, "Engineering Test Satellite VII Flight Experiments for Space Robot Dynamics and Control: Theories on Laboratory Test Beds Ten Years Ago, Now in Orbit," *The International Journal of Robotics Research*, vol. 22, no. 5, pp. 321–335, may 2003.

[12] C. W. Wampler, "Manipulator Inverse Kinematic Solutions Based on Vector Formulations and Damped Least-Squares Methods," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 16, no. 1, pp. 93–101, jan 1986.

[13] S. Chiaverini, G. Oriolo, and A. A. Maciejewski, "Redundant Robots," in *Springer Handbook of Robotics*, 2nd ed., B. Siciliano and O. Khatib, Eds. Springer International Publishing, 2016, ch. 10, pp. 221–242.

[14] S. Chiaverini, O. Egeland, and R. K. Kanestrøm, "Achieving user-defined accuracy with damped least-squares inverse kinematics," in *Fifth International Conference on Advanced Robotics Robots in Unstructured Environments*, 1991, pp. 672–677.

[15] E. Papadopoulos and S. Dubowsky, "Dynamic Singularities in Free-Floating Space Manipulators," in *Dynamic Systems, Measurement and Control*, 1993, ch. 4th.

[16] Y. K. Thong, M. S. Woolfson, J. A. Crowe, B. R. Hayes-Gill, and D. A. Jones, "Numerical double integration of acceleration measurements in noise," *Measurement: Journal of the International Measurement Confederation*, vol. 36, no. 1, pp. 73–92, 2004.

[17] J. C. Kinsey, R. M. Eustice, and L. L. Whitcomb, "A survey of underwater vehicle navigation: recent advances and new challenges," *Proc. of MCMC, Lisbon, 2006*, no. May, 2006.

[18] R. M. Eustice, H. Singh, J. Leonard, M. Walter, and R. Ballard, "Visually Navigating the RMS Titanic with SLAM Information Filters," *Robotics: Science and Systems*, pp. 57–64, 2005.

[19] P. Chiacchio, S. Chiaverini, L. Sciavicco, and B. Siciliano, "Closed-Loop Inverse Kinematics Schemes for Constrained Redundant Manipulators with Task Space Augmentation and Task Priority Strategy," *The International Journal of Robotics Research*, vol. 10, pp. 410–425, 1991.

[20] MathWorks Inc., "MATLAB Robotics System Toolbox Release 2017a," Natick, MA, 2017.

[21] CM Labs Simulations Inc., "Vortex Studio Release 6.8.1," Montreal, Canada, 2016.

[22] F. Dukan, "ROV Motion Control Systems," Ph.D. dissertation, Norwegian University of Science and Technology, 2014.

[23] P. Cieslak, P. Ridao, and M. Giergiel, "Autonomous underwater panel operation by GIRONA500 UVMS: A practical approach to autonomous underwater manipulation," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 529–536, may 2015.

References

[1] J. Yuh, “Design and control of autonomous underwater robots: A survey,” *Autonomous Robots*, vol. 8, no. 1, pp. 7–24, 2000.

[2] J. Yuh, G. Marani, and D. R. Blidberg, “Applications of marine robotic vehicles,” *Intelligent Service Robotics*, vol. 4, no. 4, pp. 221–231, Oct. 2011.

[3] J. Sverdrup-Thygeson, E. Kelasidi, K. Y. Pettersen, and J. T. Gravdahl, “The Underwater Swimming Manipulator – A Bio-Inspired AUV,” in *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*, Tokyo, Japan: IEEE, 2016, pp. 387–395.

[4] E. Kelasidi, P. Liljebäck, K. Pettersen, and J. Gravdahl, “Biologically Inspired Swimming Snake Robots: Modeling, Control and Experimental Investigation,” *IEEE Robotics and Automation Magazine*, 2015.

[5] Y. Umetani and K. Yoshida, “Resolved Motion Rate Control of Space Manipulators with Generalized Jacobian Matrix,” *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 303–314, Jun. 1989.

[6] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control.* Wiley, Apr. 2011, p. 575.

[7] P. J. From, J. T. Gravdahl, and K. Y. Pettersen, *Vehicle-Manipulator Systems*, 1st, ser. Advances in Industrial Control. London: Springer, 2014, p. 388.

[8] J. Sverdrup-Thygeson, E. Kelasidi, K. Y. Pettersen, and J. T. Gravdahl, “A control framework for biologically inspired underwater swimming manipulators

equipped with thrusters," in *Proc. 10th IFAC Conference on Control Applications in Marine Systems (CAMS)*, vol. 49, Trondheim, Norway, 2016, pp. 89–96.

[9] ——, "Modeling of underwater swimming manipulators," in *Proc. 10th IFAC Conference on Control Applications in Marine Systems (CAMS)*, vol. 49, Trondheim, Norway, 2016, pp. 81–88.

[10] E. Kelasidi and K. Y. Pettersen, "Modeling of underwater snake robots," in *IEEE International Conference on Robotics and Automation*, Hong Kong, China: IEEE, 2014, pp. 4540–4547.

[11] P. Liljebäck, K. Y. Pettersen, Ø. Stavdahl, and J. T. Gravdahl, *Snake robots: modelling, mechatronics, and control*, ser. Advances in Industrial Control. Springer London, 2012.

[12] B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial," *Journal of Intelligent and Robotic Systems*, vol. 3, no. 3, pp. 201–212, 1990.

[13] L. Sciavicco and B. Siciliano, "Solving the Inverse Kinematic Problem for Robotic Manipulators," in *RoManSy 6*, Boston, MA: Springer US, 1987, pp. 107–114.

[14] D. Whitney, "Resolved Motion Rate Control of Manipulators and Human Prostheses," *IEEE Transactions on Man Machine Systems*, vol. 10, no. 2, pp. 47–53, 1969.

[15] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-Priority Based Redundancy Control of Robot Manipulators," *The International Journal of Robotics Research*, vol. 6, no. 1, pp. 32–42, 1987.

[16] G. Antonelli and S. Chiaverini, "Task-priority redundancy resolution for underwater vehicle-manipulator systems," in *Proceedings 1998 IEEE International Conference on Robotics and Automation*, IEEE, vol. 1, Leuven, Belgium, 1998, pp. 768–773.

[17] Y. Nakamura and H. Hanafusa, "Inverse Kinematic Solutions With Singularity Robustness for Robot Manipulator Control," *Journal of Dynamic Systems, Measurement, and Control*, vol. 108, no. 3, p. 163, 1986.

[18] C. W. Wampler, "Manipulator Inverse Kinematic Solutions Based on Vector Formulations and Damped Least-Squares Methods," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 16, no. 1, pp. 93–101, Jan. 1986.

[19] A. A. Maciejewski and C. A. Klein, "Numerical filtering for the operation of robotic manipulators through kinematically singular configurations," *Journal of Robotic Systems*, vol. 5, no. 6, pp. 527–552, 1988.

[20] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 3, pp. 398–410, 1997.

[21] G. S. Chirikjian and J. W. Burdick, "A Modal Approach to Hyper-Redundant Manipulator Kinematics," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 3, pp. 343–354, 1994.

[22] R. W. Longman, R. E. Lindberg, and M. F. Zedd, "Satellite-Mounted Robot Manipulators—New Kinematics and Reaction Moment Compensation," *International Journal of Robotics Research*, vol. 6, no. 3, pp. 87–103, 1987.

[23] Y. Umetani and K. Yoshida, "Continuous path control of space manipulators mounted on OMV," *Acta Astronautica*, vol. 15, no. 12, pp. 981–986, Dec. 1987.

[24] K. Yoshida, "Engineering Test Satellite VII Flight Experiments for Space Robot Dynamics and Control: Theories on Laboratory Test Beds Ten Years Ago, Now in Orbit," *The International Journal of Robotics Research*, vol. 22, no. 5, pp. 321–335, May 2003.

[25] Z. Vafa and S. Dubowsky, "On the dynamics of manipulators in space using the virtual manipulator approach," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 4, Institute of Electrical and Electronics Engineers, 1987, pp. 579–585.

[26] K. Yoshida, B. Wilcox, G. Hirzinger, and R. Lampariello, "Space Robotics," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., 2nd, Springer International Publishing, 2016, ch. 55, pp. 1423–1462.

[27] S. Dubowsky and E. Papadopoulos, “The Kinematics, Dynamics, and Control of Free-Flying and Free-Floating Space Robotic Systems,” *IEEE Transactions on Robotics and Automation*, vol. 9, no. 5, pp. 531–543, 1993.

[28] O. Egeland and J. R. Sagli, “Coordination of Motion in a Spacecraft/Manipulator System,” *The International Journal of Robotics Research*, vol. 12, no. 4, pp. 366–379, Aug. 1993.

[29] B. Siciliano, “Closed-loop inverse kinematics algorithms for redundant spacecraft/manipulator systems,” in *Proceedings IEEE International Conference on Robotics and Automation*, IEEE Comput. Soc. Press, 1993, pp. 95–100.

[30] Y. Xu and T. Kanade, *Space robotics: dynamics and control.* New York: Springer Science & Business Media, 1993, p. 290.

[31] M. Oda, “Coordinated control of spacecraft attitude and its manipulator,” in *IEEE International Conference on Robotics and Automation*, vol. 1, Minneapolis, MN: IEEE, 1996, pp. 732–738.

[32] D. N. Nenchev, K. Yoshida, and M. Uchiyama, “Reaction Null-space Based Control of Flexible Structure Mounted Manipulator Systems,” in *Proceedings of 35th IEEE Conference on Decision and Control*, vol. 4, Kobe, Japan: IEEE, 1996, pp. 4118–4123.

[33] Y. Umetani and K. Yoshida, “Workspace and Manipulability Analysis of Space Manipulator,” *Transactions of the Society of Instrument and Control Engineers*, vol. E-1, no. 1, pp. 1–8, 2001.

[34] T. W. McLain, S. M. Rock, and M. J. Lee, “Experiments in the coordination of underwater manipulator and vehicle control,” in *’Challenges of Our Changing Global Environment’. Conference Proceedings. OCEANS ’95 MTS/IEEE*, vol. 2, San Diego, CA: IEEE, 1995, pp. 1208–1215.

[35] T. W. McLain and S. M. Rock, “Experiments in the hydrodynamic modeling of an underwater manipulator,” *Proceedings of Symposium on Autonomous Underwater Vehicle Technology*, pp. 463–469, 1996.

[36] K. J. Waldron and J. Schmiedeler, “Kinematics,” in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., 2nd, Springer International Publishing, 2016, ch. 2, pp. 11–36.

[37] O. Egeland and J. T. Gravdahl, *Modeling and Simulation for Automatic Control.* Trondheim, Norway: Marine Cybernetics, 2002, p. 639.

[38] T. Sugihara, “Solvability-unconcerned inverse kinematics by the Levenberg-Marquardt method,” *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 984–991, 2011.

[39] R. Diankov, “Automated Construction of Robotic Manipulation Programs,” PhD thesis, Carnegie Mellon University, Robotics Institute, Aug. 2010.

[40] G. Antonelli, *Underwater Robots*, 3rd, ser. Springer Tracts in Advanced Robotics. Springer International Publishing, 2014, vol. 96, p. 279.

[41] K. Yoshida and Y. Umetani, “Control of Space Manipulators with Generalized Jacobian Matrix,” in *Space robotics: Dynamics and control*, Y. Xu and T. Kanade, Eds., Springer, 1993, ch. 7, pp. 165–204.

[42] H. D. Young and R. A. Freedman, *University Physics with Modern Physics*, 14th ed., J. Zalesky, Ed. Pearson, 2016, p. 1600.

[43] T. L. Heath, *The works of Archimedes.* Cambridge University Press, 1897, p. 524.

[44] F. Dukan, “ROV Motion Control Systems,” PhD thesis, Norwegian University of Science and Technology, 2014, p. 203.

[45] M. Hildebrandt, L. Christensen, J. Kerdels, J. Albiez, and F. Kirchner, “Realtime motion compensation for ROV-based tele-operated underwater manipulators,” in *OCEANS 2009-EUROPE*, May 2009, pp. 1–6.

[46] Y. K. Thong, M. S. Woolfson, J. A. Crowe, B. R. Hayes-Gill, and D. A. Jones, “Numerical double integration of acceleration measurements in noise,” *Measurement:*

Journal of the International Measurement Confederation, vol. 36, no. 1, pp. 73–92, 2004.

[47] H.-T. Choi and J. Yuh, "Underwater Robots," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., 2nd, Springer International Publishing, 2016, ch. 25, pp. 595–622.

[48] J. C. Kinsey, R. M. Eustice, and L. L. Whitcomb, "A survey of underwater vehicle navigation: recent advances and new challenges," *Proc. of MCMC, Lisbon, 2006*, no. May, 2006.

[49] R. M. Eustice, H. Singh, J. Leonard, M. Walter, and R. Ballard, "Visually Navigating the RMS Titanic with SLAM Information Filters," *Robotics: Science and Systems*, pp. 57–64, 2005.

[50] P. Chiacchio, S. Chiaverini, L. Sciavicco, and B. Siciliano, "Closed-Loop Inverse Kinematics Schemes for Constrained Redundant Manipulators with Task Space Augmentation and Task Priority Strategy," *The International Journal of Robotics Research*, vol. 10, pp. 410–425, 1991.

[51] S. Chiaverini, G. Oriolo, and A. A. Maciejewski, "Redundant Robots," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., 2nd, Springer International Publishing, 2016, ch. 10, pp. 221–242.

[52] O.-E. Fjellstad and T. I. Fossen, "Quaternion feedback regulation of underwater vehicles," *Proceedings of IEEE International Conference on Control and Applications CCA-94*, 857–862 vol.2, 1994.

[53] MathWorks Inc., *MATLAB Robotics System Toolbox Release 2017a*, Natick, MA, 2017.

[54] R. Penrose and J. A. Todd, "A generalized inverse for matrices," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 51, no. 03, p. 406, 1955.

[55] S. Chiaverini, O. Egeland, and R. K. Kanestrøm, “Achieving user-defined accuracy with damped least-squares inverse kinematics,” in *Fifth International Conference on Advanced Robotics Robots in Unstructured Environments*, 1991, pp. 672–677.

[56] E. Papadopoulos and S. Dubowsky, “Dynamic Singularities in Free-Floating Space Manipulators,” in *Dynamic Systems, Measurement and Control*, 1993, ch. 4th.

[57] T. F. Chan and R. V. Dubey, “A Weighted Least-Norm Solution Based Scheme for Avoiding Joint Limits for Redundant Joint Manipulators,” *IEEE Transactions on Robotics and Automation*, vol. 11, no. 2, pp. 286–292, 1995.

[58] CM Labs Simulations Inc., *Vortex Studio Release 6.8.1*, Montreal, Canada, 2016.

[59] P. Cieslak, P. Ridao, and M. Giergiel, “Autonomous underwater panel operation by GIRONA500 UVMS: A practical approach to autonomous underwater manipulation,” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 529–536, May 2015.

[60] The Society of Naval Architects and Marine Engineers, *Nomenclature for Treating the Motion of a Submerged Body Through a Fluid*, 1950.

[61] T. Yoshikawa, “Measure of Manipulability for Robot Manipulators,” *Journal of the Robotics Society of Japan*, vol. 2, no. 1, pp. 63–67, 1984.

[62] D. N. Nenchev, Y. Umetani, and K. Yoshida, “Analysis of a Redundant Free-Flying Spacecraft/Manipulator System,” *IEEE Transactions on Robotics and Automation*, vol. 8, no. 1, pp. 1–6, 1992.