Appeared in *IEEE Transactions on Systems, Man, and Cybernetics*, v.25, n.8,
August 1995, pp. 1194–1206.

Efficient Dynamic Simulation of an Underwater Vehicle with a Robotic Manipulator †

Scott McMillan David E. Orin Robert B. McGhee

Abstract

In this paper, an efficient dynamic simulation algorithm is developed for an underwater robotic vehicle (URV) with a manipulator. It is based on previous work on efficient O(N) algorithms, where N is the number of links in the manipulator, and has been extended to include the effects of a mobile base (the URV body). In addition, the various hydrodynamic forces exerted on these systems in underwater environments are also incorporated into the simulation. The effects modeled in this work are added mass, viscous drag, fluid acceleration, and buoyancy forces. With efficient implementation of the resulting algorithm, the amount of computation with inclusion of the hydrodynamics is almost double that of the original algorithm for a six degree-of-freedom land-based manipulator with a mobile base. Nevertheless, the amount of computation still only grows linearly with the number of degrees of freedom in the manipulator.

---

† This work was supported in part by a DuPont Fellowship and an AT&T Ph.D. Scholarship both at The Ohio State University, by Grant No. BCS-9109989 from the National Science Foundation to the Naval Postgraduate School, and by Grant No. BCS-9311269 from the National Science Foundation to The Ohio State University.

D. E. Orin is with the Department of Electrical Engineering, The Ohio State University, Columbus, OH 43210. S. McMillan and R. B. McGhee are with the Department of Computer Science, Naval Postgraduate School, Monterey, CA 93943.

1. Introduction

The importance of underwater robotic vehicles (URVs) for marine research and subsea development continues to grow because their manned counterparts are much more expensive to develop and maintain [1, 2, 3]. This increase in use has brought about a concomitant need for accurate simulations of these systems [4], and with the addition of robotic manipulators to these vehicles, such simulations must become more sophisticated. As with land- and space-based robotics, accurate dynamic simulation can be a very beneficial tool in development of URVs. With proper use, significant amounts of time and money can be saved during the design, test and evaluation phases of new vehicle systems or subsystems. Specifically, dynamic simulation can reduce the need for costly prototypes by eliminating many candidate designs early in the development process. This saves not only on cost, but also on the time needed to construct successive prototype generations. In addition, simulators can aid in the design of control algorithms for these systems. By using the simulated URV to test such algorithms, the possibility of potentially damaging instabilities due to algorithm errors is eliminated, and risks encountered when the control system is finally implemented in hardware are reduced.

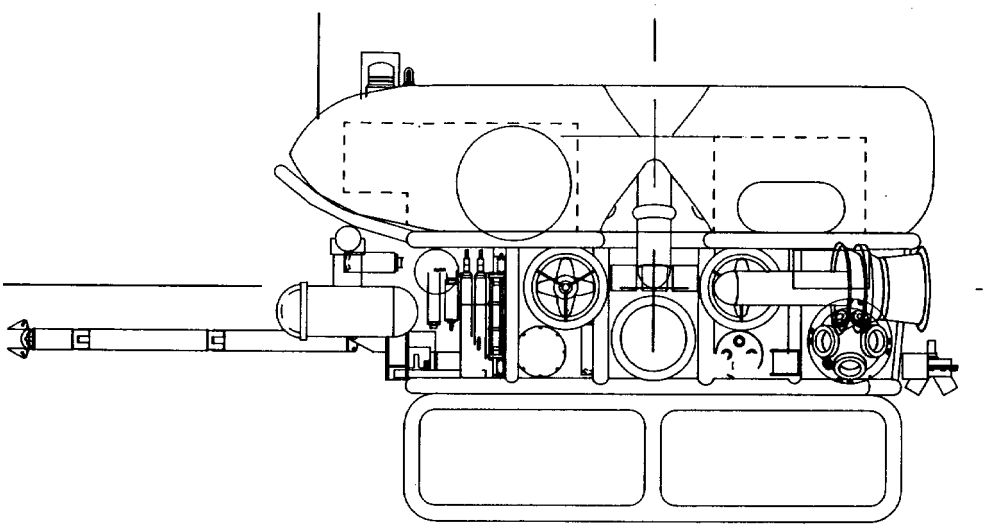
With real-time simulation rates, a number of other uses for dynamic simulators are possible. The first is hardware-in-the-loop simulation where control system hardware and software is tested by interfacing it to a real-time simulation of the URV [5]. Human-in-the-loop applications can also be implemented when a real-time simulation is coupled with a realistic 3-D graphical display of the system. One such application is to train pilots and mission specialists much like aircraft simulators are used to train aviators. Another application can be found in the teleoperation of untethered URVs. Because significant delays occur in acoustic communication with such vehicles, human control is significantly degraded. By providing the teleoperator with a simulated

display of the system, on-line with no delay, enhanced performance of human-machine interaction can be realized [6, 7].

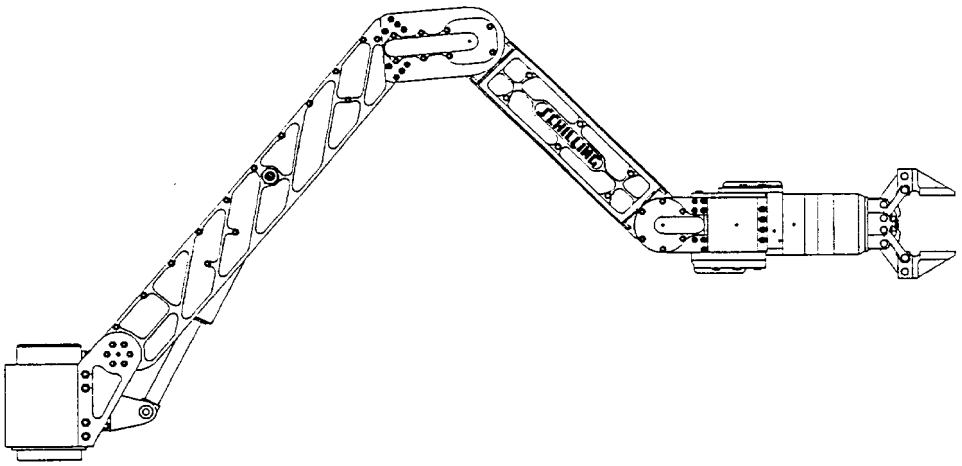
The system targeted in our research is a new remotely-operated vehicle (ROV), *Tiburon*, that is currently under development at the Monterey Bay Aquarium Research Institute (MBARI). This vehicle is designed for operation at depths of up to 4000 meters [8]. A CAD drawing of it is shown in Figure 1 along with the Schilling Titan II manipulator [9] which will be mounted on the front.

In order to achieve real-time simulation rates, efficient dynamic and hydrodynamic algorithms for systems like this must be developed. A number of efficient algorithms have been developed to compute the dynamics of the more common land-based manipulators. The two most notable approaches to this problem are the Composite Rigid Body (CRB) method [10] and the Articulated-Body (AB) method [11]. The latter has an advantage because its computation grows linearly with the number of degrees of freedom (DOFs), whereas the CRB method has cubic complexity. As a result, an efficient implementation of the AB method has been shown to require less computation than that of the CRB method for manipulators with more than three DOFs [12]. Because the Schilling arm has six DOFs, the AB simulation algorithm is more efficient than the CRB algorithm, and is the basis for the algorithm developed in our work.

The purpose of this paper is to present the development of this efficient O(N) algorithm for dynamic and hydrodynamic simulation of a URV system with a manipulator. To accomplish this, the effects of a mobile base (the URV body) must first be included. After this step, the resulting algorithm could be used in the simulation of space-based robotic systems. Then, the most efficient method for including hydrodynamic effects must be determined. To this end, work by Yuh [13] and Ioi and Itoh [14] have identified the most important forces which include added mass, viscous drag, fluid acceleration, and buoyancy. This paper will develop a method for computing



(a)



(b)

Figure 1: The MBARI URV system: (a) *Tiburon*, and (b) the Schilling *Titan II* manipulator.

these terms in the context of an articulated linkage system, and more importantly, will efficiently incorporate these into the AB simulation algorithm.

In the next section, our dynamics notation is presented along with the salient features of the AB algorithm and a synopsis of the principal equations. The basic AB algorithm is also extended to include the dynamics of a mobile base. In Section 3, the hydrodynamic terms needed in the simulation of a single rigid body are presented. Then, the AB algorithm is extended in Section 4 to include these hydrodynamic terms. The computational requirements for the resulting algorithm are presented in Section 5, along with a comparison of algorithms for the fixed- and mobile-base systems on land or in space. Using object-oriented design techniques, this algorithm has been implemented in C++. Computational runtimes using this software package, called **DynaMechs** [15], are presented in Section 6.

2. Articulated-Body Dynamics: Background and Notation

In this section, Featherstone's AB method [11] for computing robot dynamics is reviewed, and the notation used in this paper is presented. This is a very efficient method for computing the dynamics of multibody systems [16]. To begin this discussion, a serial chain manipulator with a fixed base as shown in Figure 2 is assumed. It has N links that are numbered from 1, attached to the base through joint 1, to N, the end-effector. The joints between the links can have an arbitrary number of DOFs, but we will limit this discussion to single DOF revolute or prismatic joints. This assumption accounts for the vast majority of robotic systems while simplifying the analysis of computational requirements. Nevertheless, this is not limiting, since multiple DOF joints can be simulated by concatenation of multiple single DOF joints.

Each joint axis of the manipulator, as indicated by the arrows in the figure, is

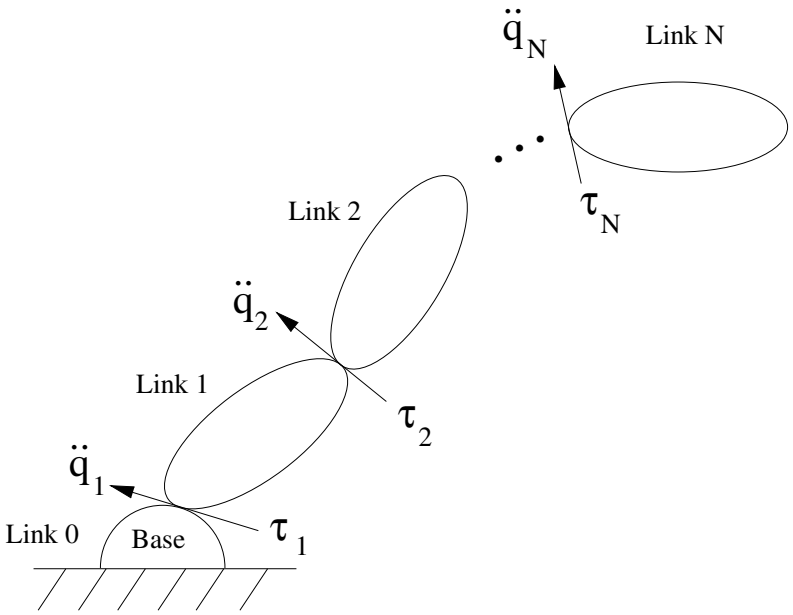


Figure 2: Serial-chain manipulator.

specified by a six-element unit vector, $\boldsymbol{\phi}_i$, where the subscript i indicates the joint or link number. These vectors are part of the spatial notation used in our work that combines three-dimensional angular and translational quantities into a single vector [17, 18]. Since $\boldsymbol{\phi}_i$ is defined in the coordinate system fixed to link i, it is constant. Using modified Denavit-Hartenberg (MDH) notation [19, 20], in which the z-axis lies along the ith joint axis, $\boldsymbol{\phi}_i$ is given by $[0\ 0\ 1\ 0\ 0\ 0]^T$ for revolute joints and $[0\ 0\ 0\ 0\ 0\ 1]^T$ for prismatic joints.

The state of the system and its inputs can now be specified with a set of scalars that are defined with respect to these vectors as shown in Figure 2. For each joint i, the state is given by the scalar joint position and velocity, q_i and $\dot{q}_i$, and the input joint torque or force is given by τ_i. Given these quantities for all of the joints, the goal of dynamic simulation is to compute the forward dynamics for the N joint accelerations, $\ddot{q}_i$, and then to numerically integrate these to obtain new values for the positions and

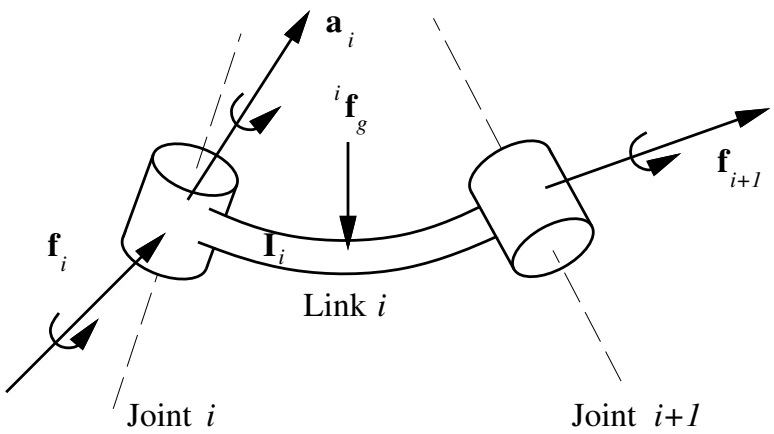


Figure 3: Spatial notation for rigid-body dynamics.

velocities. In this section, the algorithm to compute forward dynamics is described. For a discussion of the various numerical integration algorithms for solving ordinary differential equations, the reader is referred to any of a number of numerical analysis texts (e.g., [21]).

A. Dynamics

Derivation of the AB algorithm begins with the set of dynamics equations for the force balance on each link. As illustrated in Figure 3, the force balance equation for link i operating in air or on the surface of the Earth is given as follows:

$$\mathbf{f}_i - {}^{i+1}\mathbf{X}_i^T \mathbf{f}_{i+1} + {}^i\mathbf{f}_g \;=\; \mathbf{I}_i \mathbf{a}_i - \boldsymbol{\beta}_i, \tag{1}$$

where $\mathbf{f}_i$ is the spatial force exerted onto link i by its inboard link and contains the effect of τ_i, $\mathbf{f}_{i+1}$ is the force exerted by link i onto the next link outboard and contains the effect of τ_{i+1}, and ${}^i\mathbf{f}_g$ is the gravitational force. These are six-element vectors combining the three-dimensional moment, $\boldsymbol{n}$, and translational force, $\boldsymbol{f}$, vectors as

follows:

$$\mathbf{f}_i = \begin{bmatrix} \boldsymbol{n}_i \\ \boldsymbol{f}_i \end{bmatrix}. \tag{2}$$

Like the spatial forces, the spatial acceleration of the link, $\mathbf{a}_i$, is also a six-element vector combining the angular, $\dot{\boldsymbol{\omega}}_i$, and translational, $\boldsymbol{a}_i$, acceleration vectors*.

The spatial transformation matrix, ${}^{i+1}\mathbf{X}_i$, is used to transform spatial vectors between coordinate systems i and $i+1$ and is defined as follows [20]:

$${}^{i+1}\mathbf{X}_i = \begin{bmatrix} {}^{i+1}\boldsymbol{R}_i & \mathbf{0} \\ {}^{i+1}\boldsymbol{R}_i\,{}^{i}\tilde{\boldsymbol{p}}_{i+1}^{T} & {}^{i+1}\boldsymbol{R}_i \end{bmatrix}, \tag{3}$$

where ${}^{i+1}\boldsymbol{R}_i$ is the 3×3 rotation matrix from the coordinate system attached to link i to the one attached to link $i+1$, and ${}^{i}\boldsymbol{p}_{i+1}$ is the Cartesian vector specifying the position of the origin of link $(i+1)$'s coordinate system with respect to link i's. The tilde above the vector signifies that its components should be combined in a skew symmetric matrix such that $\tilde{\boldsymbol{b}}\boldsymbol{a} = \boldsymbol{b} \times \boldsymbol{a}$.

In the first term on the right side of Eq. (1), link i's spatial inertia, $\mathbf{I}_i$, relates the spatial acceleration of the link to the resultant spatial force. This 6×6 inertia matrix combines the link's mass and inertia quantities (first and second mass moments) as follows:

$$\mathbf{I}_i = \begin{bmatrix} \bar{\boldsymbol{I}}_i & \tilde{\boldsymbol{h}}_i \\ \tilde{\boldsymbol{h}}_i^{T} & m_i \mathbf{1}_3 \end{bmatrix}, \tag{4}$$

where $\bar{\boldsymbol{I}}_i$ is the 3×3 moment of inertia tensor for the link with respect to its own coordinate system, and $\mathbf{1}_3$ is the 3×3 identity matrix. The 3×1 vector, $\boldsymbol{h}_i$, is its first mass moment which is equal to $(m_i\,\boldsymbol{s}_i)$, where m_i is its mass and $\boldsymbol{s}_i$ is the vector from

*Note that the convention in this paper is to use an italic bold variable, such as $\boldsymbol{f}$, to refer to a Cartesian (three-dimensional) vector representing either a rotational or translational quantity, and a block bold variable, such as $\mathbf{f}$, to refer to a spatial (six-dimensional) vector.

the link's coordinate system origin origin to its center of mass. Finally, in Eq. (1), $\boldsymbol{\beta}_i$ is the vector of velocity-dependent bias forces [22] (see Table 1).

In efficient robotic dynamics algorithms for ground, air, or space vehicles, the gravitational effects can be combined with the spatial acceleration in Eq. (1) to reduce the amount of computation. This is accomplished with the following force balance equation:

$$\mathbf{f}_i - {}^{i+1}\mathbf{X}_i^T \mathbf{f}_{i+1} = \mathbf{I}_i \mathbf{a}'_i - \boldsymbol{\beta}_i, \tag{5}$$

where

$$\mathbf{a}'_i = \mathbf{a}_i - \begin{bmatrix} \mathbf{0} \\ {}^i\boldsymbol{a}_g \end{bmatrix}, \tag{6}$$

which is the spatial acceleration of the body biased by gravitational acceleration.

B. Kinematics

To complete the derivation of the AB algorithm, a set of kinematic equations that compute the angular and translational accelerations of each link given the corresponding accelerations of the inboard link and the relative joint acceleration between the two links is needed. The same equations can also be used to propagate the computation of the biased acceleration. Using spatial notation, the equation for the biased acceleration of link i (at its coordinate system) may be written as follows [17]:

$$\mathbf{a}'_i = {}^i\mathbf{X}_{i-1}\, \mathbf{a}'_{i-1} + \boldsymbol{\phi}_i \ddot{q}_i + \boldsymbol{\zeta}_i, \tag{7}$$

where $\boldsymbol{\zeta}_i$ is the vector of Coriolis and centripetal accelerations that are a function of known link velocities (see Table 1).

From Eq. (7), the acceleration of link i is dependent on the acceleration of the

inboard link. This implies that a recursion[†] from the base of the manipulator to the end-effector may be used to compute the link accelerations when the joint accelerations, $\ddot{q}_i$, are known. Eq. (1) implies a recursion from the end-effector to the base to determine all of the link forces once the biased accelerations have been determined. Together these two sets of equations define the outward and inward recursions associated with the *inverse* dynamics problem which computes the joint torques for a given motion. For the case of forward dynamics computations, however, the joint accelerations are unknown so that the biased accelerations, and hence the link forces cannot be determined in a direct manner from these equations. Consequently, a different approach is required to solve the dynamic simulation problem. In the AB algorithm, this involves the computation of AB inertias.

C. Articulated Bodies

Instead of using the force balance equation for a single link, an expression relating $\mathbf{f}_i$ to the dynamic properties of links i through N is used. As illustrated in Figure 4, this relationship is given as follows:

$$\mathbf{f}_i = \mathbf{I}_i^* \mathbf{a}_i' - \boldsymbol{\beta}_i^*. \tag{8}$$

The matrix, $\mathbf{I}_i^*$, is the 6×6 AB inertia of links i through N which is the inertia that is "felt" at the ith coordinate system when the joints from $i+1$ to N are free to move. Likewise, the vector, $\boldsymbol{\beta}_i^*$, is the bias force exerted on the ith link due to the resultant bias forces within the articulated body including all outboard joint torques

[†] This is the term generally used in the robotics literature. However, for those not familiar with this literature, its usage should not be confused with the notion of *recursive function calls* in computer science terminology. In the context of robotics, this term does not imply that the evaluation of this equation is implemented with recursive function calls from end-effector to base (although it could be). Rather, it is implemented with a `for`-loop and each acceleration is computed in succession from base to end-effector.

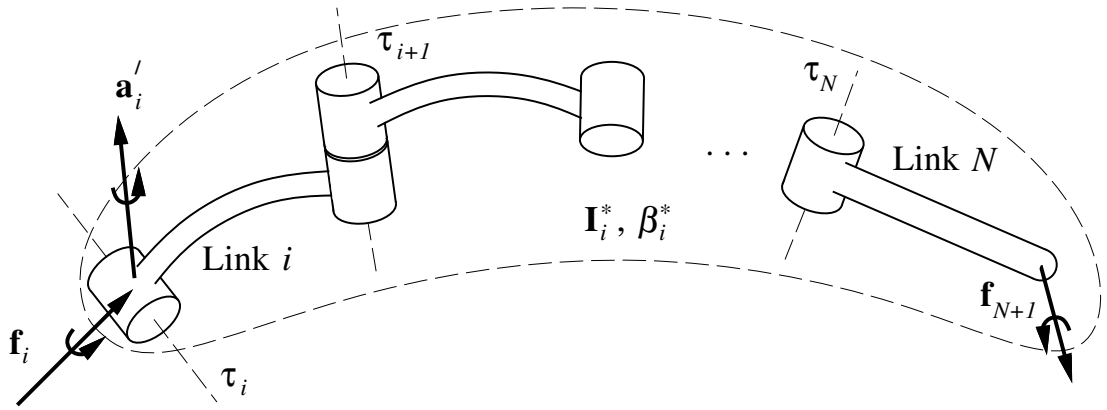


Figure 4: Articulated-body dynamics.

and excluding gravitational effects. Details of the derivation of the equations needed to efficiently compute $\mathbf{I}_i^*$ and $\boldsymbol{\beta}_i^*$ can be found in [12].

The resulting AB algorithm for forward dynamics contains three O(N) recursions. The first is a Forward Kinematics recursion which computes the velocities and velocity-dependent terms, $\boldsymbol{\beta}_i$ and $\boldsymbol{\zeta}_i$, of each link from the base to the tip. In the second step, the AB inertias, $\mathbf{I}_i^*$, and bias forces, $\boldsymbol{\beta}_i^*$, are computed in a Backward Dynamics recursion from the tip to the base. The final step begins with the known base acceleration, $\mathbf{a}_0 = \mathbf{0}$, which enables the computation of the first joint's acceleration, $\ddot{q}_1$, with an equation derived from Eqs. (7) and (8). This enables the computation of link 1's acceleration from Eq. (7). These results are used to compute the joint and link accelerations (in that order) for the next link in the chain. This procedure defines the final Forward Accelerations recursion from the base to the tip of the chain.

D. Mobile Base

Thus far, we have been discussing an AB simulation algorithm for a serial chain with a fixed base which can be found in [11, 22]. When simulating URV systems, the base of the manipulator, the vehicle body, is no longer fixed with respect to an

inertial frame and the equations must be augmented to model this characteristic. This is accomplished by modeling the vehicle body as another link (link 0) in the serial chain which has a six DOF joint (joint 0) between it and the inertial frame, and adding another step to each of the three recursions. The spatial representation of this joint's motion, $\boldsymbol{\phi}_0$, is given by a constant 6×6 identity matrix. This implies that the joint velocity is equal to the spatial velocity of the base, $\dot{\mathbf{q}}_0 = \mathbf{v}_0$, expressed in body fixed coordinates. Likewise, the joint acceleration is the same as its spatial acceleration, $\ddot{\mathbf{q}}_0 = \mathbf{a}_0$, and the joint torque/force is equal to external spatial forces, $\boldsymbol{\tau}_0 = \mathbf{f}_0$, exerted on the base such as the resultant thruster force.

The Forward Kinematics recursion will now begin with computation of velocity-dependent terms for the mobile base. This is simplified because the velocity of this body is given as part of the system state and does not need to be computed. In addition, the bias acceleration, $\boldsymbol{\zeta}_0$, is zero. Then, the Backward Dynamics recursion is extended to include the computation of the AB inertia and bias force, $\mathbf{I}_0^*$ and $\boldsymbol{\beta}_0^*$, for this body.

Finally, the Forward Accelerations recursion must now begin with the computation of the base acceleration. Substituting $\boldsymbol{\phi}_0$ into the equations used to compute the joint accelerations of each link, a simplified equation which was derived by Featherstone ([17], p. 122) results:

$$\mathbf{a}_0 \;=\; (\mathbf{I}_0^*)^{-1}\,(\mathbf{f}_0 + \boldsymbol{\beta}_0^{*'}), \tag{9}$$

where $\mathbf{I}_0^*$ is the AB inertia of the entire URV system, including added mass, that is "felt" at the vehicle body's coordinate system. The net force acting on this inertia including gravitational effects is divided into two components: the resultant thruster force exerted on the vehicle, $\mathbf{f}_0$, and the rest of the forces exerted on this inertia including gravity, $\boldsymbol{\beta}_0^{*'}$. A derivation using the biased acceleration was performed in [12] which shows that this equation can be written using the biased acceleration as

follows:

$$\mathbf{a}_0' = (\mathbf{I}_0^*)^{-1}(\mathbf{f}_0 + \boldsymbol{\beta}_0^*), \tag{10}$$

where $\boldsymbol{\beta}_0^*$ now excludes gravitational effects. This biased acceleration is used as the starting condition for the final recursion along the chain to compute joint accelerations. For simulation purposes, the unbiased acceleration is needed and is obtained by adding gravitational acceleration back into this result.

3. Hydrodynamics for Rigid Bodies

When the motion of rigid bodies is to be simulated in an underwater environment, a number of additional effects must be modeled in the simulation as a result of various hydrodynamic forces. While these forces result from incompressible fluid flow determined by the Navier-Stokes (distributed fluid-flow) equations [23], "lumped" approximations to these forces are used in this work. To this end, Yuh [13] and Ioi and Itoh [14] have identified four separate effects that need to be included in a dynamic simulation of submerged rigid bodies. Under limiting assumptions that the net hydrodynamic force on an object can be represented as a sum of separately identified components modeling the effects of added mass, drag, fluid acceleration, and buoyancy forces, this section develops a notation consistent with the previous section, and derives the equations needed to compute these hydrodynamic forces exerted on a single rigid body. It is further assumed that forces computed for one link are negligibly affected by the proximity of another.

A. Added Mass

To those acquainted with the dynamics of manipulators in space or air, probably the most surprising hydrodynamic effect is the added mass force. When a body is

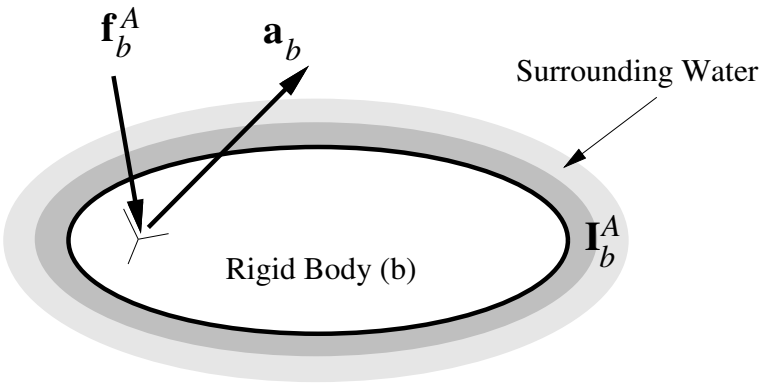


Figure 5: Added mass.

accelerated through a fluid, some of the surrounding fluid is also accelerated with the body. A force is exerted on the surrounding fluid to achieve this acceleration, and the reaction force, which is equal in magnitude and opposite in direction, is exerted on the body. The latter is referred to as the *added mass* force [24]. Its computation within our framework is presented in this section.

With our assumptions about lumped approximations, added mass can be specified with a 6×6 matrix, $\mathbf{I}_b^A$, as shown in Figure 5. As with the spatial inertia for a rigid body, this added mass matrix is symmetric and positive-definite. Since this inertia is a function of the body's surface geometry, however, there is no concept of principal axes as in rigid body analysis, along which, torque and angular momentum are colinear. In fact, with added mass, unlike the rigid body's mass, an applied translational force can result in a non-colinear acceleration of the center of gravity as well. Consequently, the added mass matrix does not have the same structure as shown in Eq. (4) for the spatial inertia of a rigid body in space or air. For a general body shape, the matrix will be full which leads to notably different dynamic behavior as compared to the rigid body counterpart.

Newman [24] derived a set of equations to compute the added mass force that is exerted on a rigid body accelerating through an unbounded, inviscid fluid that is itself

not accelerating (that is, has steady, irrotational flow). This was found by taking the derivative of the total momentum of the fluid. As shown in Appendix A, translating Newman's equations into spatial notation results in the following equation:

$$\mathbf{f}_b^A = -\mathbf{I}_b^A \begin{bmatrix} \dot{\boldsymbol{\omega}}_b \\ \dot{\boldsymbol{v}}_b \end{bmatrix} - \begin{bmatrix} \tilde{\boldsymbol{\omega}}_b & \tilde{\boldsymbol{v}}_b \\ \mathbf{0} & \tilde{\boldsymbol{\omega}}_b \end{bmatrix} \mathbf{I}_b^A \begin{bmatrix} \boldsymbol{\omega}_b \\ \boldsymbol{v}_b \end{bmatrix}, \tag{11}$$

where negative signs are needed to compute the reaction force that is exerted *onto* the rigid body, and $\boldsymbol{\omega}_b$ and $\boldsymbol{v}_b$ are the angular and translational velocities of the body, respectively. The translational velocity derivative term, $\dot{\boldsymbol{v}}_b$, is not the true acceleration of the rigid body, but is rather the time derivative of $\boldsymbol{v}_b$ with respect to the body's rotating reference frame.[‡]

Since the AB simulation algorithm uses the biased acceleration of the body, $\boldsymbol{a}'_b$, Eq. (11) must be modified before it can be efficiently incorporated into the algorithm. First, the following relationship for its true acceleration, $\boldsymbol{a}_b$, is used:

$$\boldsymbol{a}_b = \dot{\boldsymbol{v}}_b + \boldsymbol{\omega}_b \times \boldsymbol{v}_b, \tag{12}$$

where $\dot{\boldsymbol{v}}_b$ is frequently referred to as the *rate of growth* of a vector, and $\boldsymbol{\omega}_b \times \boldsymbol{v}_b$ the *rate of transport* [26]. Substituting this into Eq. (11) leads to the following equation for the added mass force:

$$\mathbf{f}_b^A = -\mathbf{I}_b^A \begin{bmatrix} \dot{\boldsymbol{\omega}}_b \\ \boldsymbol{a}_b \end{bmatrix} + \mathbf{I}_b^A \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\omega}_b \times \boldsymbol{v}_b \end{bmatrix} - \begin{bmatrix} \tilde{\boldsymbol{\omega}}_b & \tilde{\boldsymbol{v}}_b \\ \mathbf{0} & \tilde{\boldsymbol{\omega}}_b \end{bmatrix} \mathbf{I}_b^A \begin{bmatrix} \boldsymbol{\omega}_b \\ \boldsymbol{v}_b \end{bmatrix}. \tag{13}$$

Given this result, the effects of fluid translational acceleration on the force resulting from added mass can be incorporated by replacing the translational acceleration of the rigid body with its translational acceleration *relative* to the surrounding fluid, $\boldsymbol{a}_b^r$ ([24], p. 150). This is defined as follows:

$$\boldsymbol{a}_b^r = \boldsymbol{a}_b - {}^b\boldsymbol{a}_f, \tag{14}$$

[‡]The elements of which are $\dot{u}$, $\dot{v}$, and $\dot{w}$ from [25].

where ${}^b\boldsymbol{a}_f$ is the translational acceleration of the fluid expressed in the body-fixed coordinate system. Likewise, the translational velocity term is replaced in the added mass equation with the relative translational velocity, $\boldsymbol{v}_b^r$.

Finally, the equation can be written in terms of the body's biased acceleration, $\mathbf{a}'_b$, so that it may be more easily incorporated into the AB dynamics algorithm. This leads to the final form of the added mass force equation:

$$\mathbf{f}_b^A = -\mathbf{I}_b^A \mathbf{a}'_b + \boldsymbol{\beta}_b^A, \tag{15}$$

where

$$\boldsymbol{\beta}_b^A = \mathbf{I}_b^A \begin{bmatrix} \mathbf{0} \\ \left({}^b\boldsymbol{a}_f - {}^b\boldsymbol{a}_g\right) + \boldsymbol{\omega}_b \times \boldsymbol{v}_b^r \end{bmatrix} - \begin{bmatrix} \tilde{\boldsymbol{\omega}}_b & \tilde{\boldsymbol{v}}_b^r \\ \mathbf{0} & \tilde{\boldsymbol{\omega}}_b \end{bmatrix} \mathbf{I}_b^A \begin{bmatrix} \boldsymbol{\omega}_b \\ \boldsymbol{v}_b^r \end{bmatrix}, \tag{16}$$

which is called the *added mass bias force* and is a function of known state variables, fluid velocity and acceleration, and gravity.

B. Drag and Lift Forces

When an object moves through a viscous fluid, drag and lift forces are exerted on it. Since water density is significant, large viscous forces can be exerted on URV systems even for reasonably slow motions. Lift and the related forces due to vortex shedding [24] are believed to be small for the applications at hand and are ignored. Drag can be decomposed into pressure drag, which is normal to the surface of the body, and shear drag, which is tangential. For underwater manipulation, the shear drag will also typically be small, so that the emphasis here is on the modeling of the pressure drag.

Pressure drag arises from non-zero normal components of relative velocity between the body's surface and the fluid. For a general body, a surface integral over the entire body is required to compute the resultant force and moment, $\mathbf{f}_b^D$. To avoid this

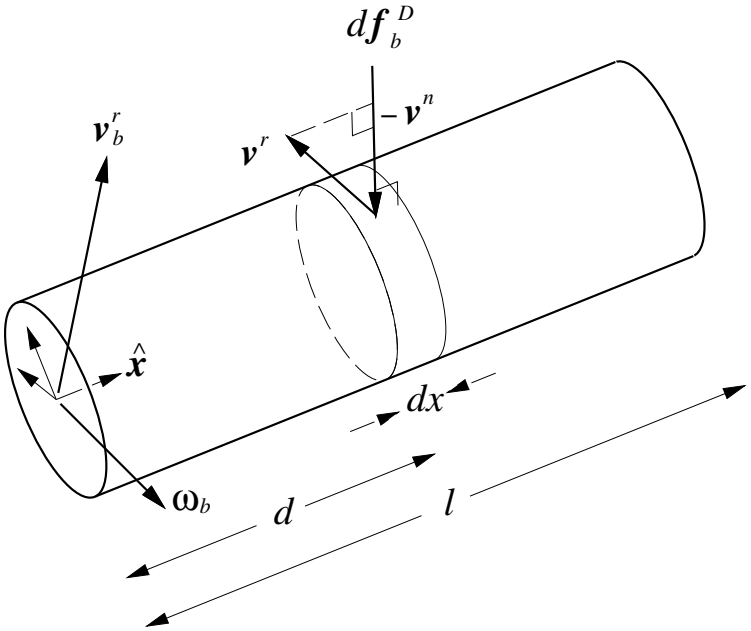


Figure 6: Drag force on a cylinder.

integration, links are approximated by cylinders as shown in Figure 6 where one of the coordinate axes is assumed to lie along the axis of the cylinder. This is not an unreasonable assumption since most often, the x- or y-axis points along links with revolute joints, and the z-axis points along links with prismatic joints when using MDH parameters to assign coordinate system locations. The resulting procedure to compute $\mathbf{f}_b^D$ is based on one in [27] which has been extended in our work to include the effects of arbitrary angular and translational velocity of the cylinder as well.

Strip theory is used to replace the surface integral with a line integral along the length of the cylinder. Therefore, the cylinder is partitioned into circular disk elements with width dx, and the translational velocity relative to the fluid and normal to the edge of each disk, $\boldsymbol{v}^n$, must be determined. The translational velocity of a disk relative to the fluid at a distance d along the cylinder's axis (the x-axis in this example) is

approximated, assuming its radius is small compared to the length, as follows:

$$\boldsymbol{v}^r(d) \;=\; \boldsymbol{v}_b^r + \boldsymbol{\omega}_b \times [d \;\; 0 \;\; 0]^T . \tag{17}$$

where $\boldsymbol{\omega}_b$ is the angular velocity of the cylinder (which is also the angular velocity relative to the fluid since it is assumed to be irrotational), and $\boldsymbol{v}_b^r$ is the translational velocity of the cylinder relative to the fluid at the origin of the body-fixed coordinate system. The normal velocity is this vector's projection onto the yz-plane.

Using the above results, the partial force exerted on the edge of a disk at a distance, d, from the coordinate system can be computed as follows:

$$d\boldsymbol{f}_b^D(d) \;=\; -0.5\rho C_D ||\boldsymbol{v}^n(d)||\; \boldsymbol{v}^n(d)\,(2r\; dx)\,, \tag{18}$$

where ρ is the fluid density, C_D is the drag coefficient, r is the radius of the cylinder, and the last term (within the parentheses) is the projected area of the disk normal to the fluid flow. The partial moment about the body-fixed coordinate system due to this force is computed as follows:

$$d\boldsymbol{n}_b^D(d) \;=\; -0.5\rho C_D ||\boldsymbol{v}^n(d)|| \left\{ [d \;\; 0 \;\; 0]^T \times \boldsymbol{v}^n(d) \right\} 2r\; dx. \tag{19}$$

Eqs. (18) and (19) must then be integrated along the length of the cylinder to obtain the resultant drag force and moment as follows:

$$\boldsymbol{f}_b^D \;=\; -\rho C_D r \int_0^l ||\boldsymbol{v}^n(x)||\boldsymbol{v}^n(x) dx \tag{20}$$

$$\boldsymbol{n}_b^D \;=\; -\rho C_D r \int_0^l ||\boldsymbol{v}^n(x)|| \left([x \;\; 0 \;\; 0]^T \times \boldsymbol{v}^n(x) \right) dx, \tag{21}$$

which make up the bottom and top parts of the spatial drag force, respectively.

The x-components of both of these vectors are zero. Therefore, for smooth cylinders, no moment about the x-axis exists which would only be caused by x-components of angular velocity. This is consistent with the earlier assumption that shear drag is negligible and assumed to be zero in this derivation. A drag force along the x-axis

can exist, however, due to a drag force exerted on the flat ends of the cylinder to relative translational velocity along the x-axis. In this case, the x-component of $\boldsymbol{f}_b^D$ is computed as follows:

$$(f_b^D)^x = -0.5\rho C_D \left(\pi r^2\right) ||(v_b^r)^x|| \, (v_b^r)^x, \tag{22}$$

where $(v_b^r)^x$ is the x-component of the link's translational velocity with respect to the fluid. The assumption that components of the drag force can be computed from normal components of relative velocity while ignoring tangential components is consistent with the work of Chakrabarti, Tam, and Wolbert [28] and the *independence principle* [27].

C. Total Buoyancy: Buoyancy and Fluid Acceleration

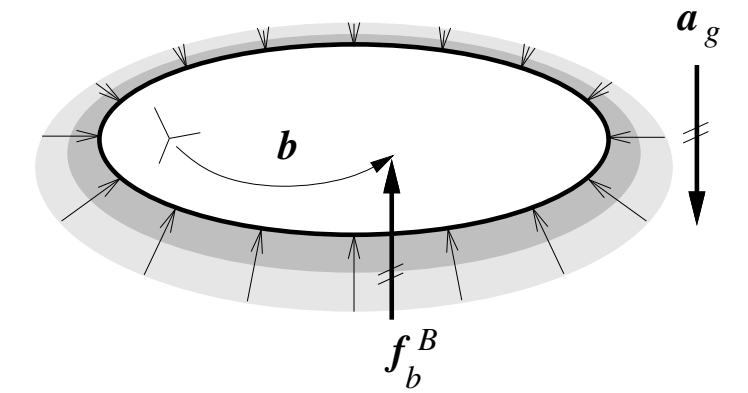
Because of the similarity between buoyancy and fluid acceleration forces, they are presented together in this section. Both are translational forces as illustrated in Figure 7. They are exerted at the center of buoyancy of the body, which is the center of volume of the body or equivalently the center of mass of the fluid that is displaced by the body. Finally, they are proportional to the mass of the fluid that is displaced by the body, m_b^f.

The buoyant force, $\boldsymbol{f}_b^B$, is exerted in the direction opposite of gravity. This force is a result of Archimedes principle which states that a body immersed in a fluid is buoyed up with a force equal to the weight of the fluid displaced by the body. In terms of the gravitational acceleration, ${}^b\boldsymbol{a}_g$, this force is computed as follows:

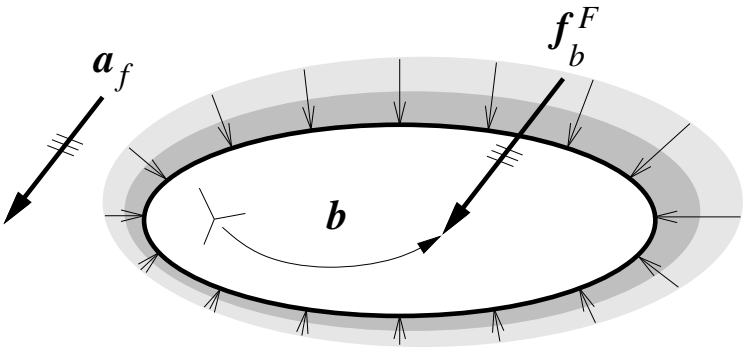
$$\boldsymbol{f}_b^B = -m_b^f \, {}^b\boldsymbol{a}_g. \tag{23}$$

The gravitational acceleration is defined as follows:

$${}^b\boldsymbol{a}_g = g \, \hat{\mathbf{z}}_e, \tag{24}$$



(a)



(b)

Figure 7: Total buoyancy forces: (a) buoyancy and (b) fluid acceleration.

where $\hat{\mathbf{z}}_e$ is the unit vector that points "down" as is traditional in marine mechanics [25], and g is the gravitational constant. A similar equation was given by Newman ([24], p. 152) for the fluid acceleration force which is given as follows:

$$\boldsymbol{f}_b^F = m_b^f \, {}^b\boldsymbol{a}_f, \tag{25}$$

where ${}^b\boldsymbol{a}_f$ is the acceleration of the fluid. This force is sometimes referred to as the *horizontal buoyancy* force.

For increased computational efficiency, both forces are combined as follows:

$$\boldsymbol{f}_b^{TB} = m_b^f \left({}^b\boldsymbol{a}_f - {}^b\boldsymbol{a}_g \right). \tag{26}$$

It is convenient to refer to this as the *total buoyancy* force. To use this in existing robot dynamics algorithms, the equivalent spatial force exerted at the origin of the body-fixed coordinate system must be found. The resulting force is computed by the following equation:

$$\mathbf{f}_b^{TB} = \begin{bmatrix} \boldsymbol{b} \times \boldsymbol{f}_b^{TB} \\ \boldsymbol{f}_b^{TB} \end{bmatrix}, \tag{27}$$

where $\boldsymbol{b}$ is the vector from the body-fixed coordinate system to its center of buoyancy. Note that the quantity, ${}^b\boldsymbol{a}_f - {}^b\boldsymbol{a}_g$, is also required in the computation of the added mass bias force, $\boldsymbol{\beta}_b^A$, and because ${}^b\boldsymbol{a}_f$ or ${}^b\boldsymbol{a}_g$ are not needed individually, the subtraction is performed once with respect to the inertial coordinate system and only a single vector needs to be transformed to each body's coordinate system.

D. Assumptions

At this point a few comments should be made about the derivation of the hydrodynamic terms and some of the assumptions that have been employed. The most notable assumptions are that the fluid is irrotational and unbounded. The former is acceptable since rotation due to any vortices in the fluid would be small compared to rotation of the body, or it would be on such a small scale compared to the extent of the body as to be negligible. The exception is wave action in shallow depths, which is not an environment that will be encountered by most URV systems. The unbounded assumption poses more of a problem, but for first order approximations it should generally be acceptable.

Another assumption is that the added mass matrix and drag coefficients are known and constant. In actuality, these quantities are extremely difficult to compute with a high degree of accuracy, and vary non-linearly with respect to velocity and other parameters [27]. However, we believe that over the range of operating conditions

typically encountered by a URV, these coefficients vary only small amounts such that a constant coefficient assumption is the only reasonable approach and is adequate for the purposes of the desired application.

Lift force is one hydrodynamic effect that has been omitted from this discussion. In most texts it is described as a force proportional to the square of the relative velocity in a direction normal to the fluid flow by some coefficient of lift C_L caused by non-zero net circulation around the body. This is usually presented with the definition for drag which is the same except that this force is in a direction opposite to the flow and related by the drag coefficient, C_D. For the three-dimensional derivations in this paper, this definition does not seem to be adequate. The line integrals that are used to compute the drag force, can, in the case of general translation, compute components normal and parallel to the fluid velocity. We have called this the computation for drag only, and considered lift forces to be present only when the body is a foil. Since the URV under consideration and the links of its robotic arm are not foils, computation of lift forces has been omitted.

Vortex shedding is another effect that has not been considered in this discussion. When a bluff body is translating through a fluid, pairs of vortices build up behind it. They continue to grow in size until instabilities cause them to alternately detach from the body. This introduces additional periodic forces on the body that can be large [29]. A straightforward check of the Strouhal Number/Reynolds Number map will indicate if vortex shedding effects are a potential problem, although precise modeling of these effects for underwater manipulators would be lengthy. Engineering solutions to eliminate the problem using spoilers could be effective, so for now the phenomenon is believed to be small enough to ignore in most robotic applications.

4. URV Simulation Algorithm Development

In this section, the dynamic simulation algorithm for an articulated chain of rigid bodies in an underwater environment is developed. First, the dynamic equation of motion for a single link in a fluid is developed. To accomplish this, the spatial force balance on a given link i is written to add the hydrodynamic forces derived in the previous section into Eq. (5). The resulting equation is written as follows:

$$\mathbf{f}_i - {}^{i+1}\mathbf{X}_i^T\ \mathbf{f}_{i+1} + \mathbf{f}_i^A + \mathbf{f}_i^D + \mathbf{f}_i^{TB} \quad = \quad \mathbf{I}_i\ \mathbf{a}_i' - \boldsymbol{\beta}_i. \tag{28}$$

Note that $\mathbf{f}_i^A$, the added mass force from Eq. (15), is also a function of, $\mathbf{a}_i'$, the unknown biased acceleration of link i. Grouping these acceleration terms together and rewriting the equation to resemble the form of Eq. (5) leads to the desired hydrodynamic equation of motion for the link:

$$\mathbf{f}_i - {}^{i+1}\mathbf{X}_i^T\ \mathbf{f}_{i+1} \quad = \quad \mathbf{I}_i^H \mathbf{a}_i' - \boldsymbol{\beta}_i^H, \tag{29}$$

where

$$\mathbf{I}_i^H \quad = \quad \mathbf{I}_i + \mathbf{I}_i^A, \quad \text{and} \tag{30}$$

$$\boldsymbol{\beta}_i^H \quad = \quad \boldsymbol{\beta}_i + \boldsymbol{\beta}_i^A + \mathbf{f}_i^D + \mathbf{f}_i^{TB}. \tag{31}$$

This equation states that the spatial acceleration of the body and the exerted force are linearly related through the sum of the rigid body's spatial inertia and the added mass of the fluid surrounding the body.

A comparison of Eq (29) with Eq. (5) leads to three changes that must be made to the AB algorithm for land-based systems to incorporate these hydrodynamic terms. First, the computation of the body's translational velocity relative to the fluid, $\boldsymbol{v}_i^r$, and the combined acceleration term, ${}^i\boldsymbol{a}_{f-g}$, for each rigid body must be added to the Forward Kinematics step. Then, the computation of the *hydrodynamic bias force*,

$\boldsymbol{\beta}_i^H$, from Eq. (31) can be performed. Finally, the link's spatial inertia, $\mathbf{I}_i$, is replaced with its *hydrodynamic inertia*, $\mathbf{I}_i^H$, from Eq. (30) in computation for the Backward Dynamics recursion. Note that the Forward Accelerations step is unaffected by the addition of hydrodynamic effects. With these modifications, an efficient implementation of the AB algorithm for URV systems with a single manipulator containing revolute ($\sigma_i = 1$) or prismatic ($\sigma_i = 0$) joints results and is listed in Table 1.

In the Forward Kinematics step, the computation of quantities for the base are separated from the recursion along the chain because of the differences with the computation for each link. In order to perform this computation, the state of the base, consisting of position, orientation and spatial velocity, and the fluid's velocity and acceleration with respect to the inertial coordinate system, ${}^e\boldsymbol{v}_f$ and ${}^e\boldsymbol{a}_f$, must be given. Given its orientation, the corresponding rotation matrix between the base and the earth-fixed inertial coordinate system, ${}^0\boldsymbol{R}_e$ is specified, and the gravitational acceleration, ${}^0\boldsymbol{a}_g$, can be determined. From these, $\boldsymbol{v}_0^r$ and ${}^0\boldsymbol{a}_{f-g}$ can be computed for the base.

For the forward recursion along the chain, the manipulator's state, including joint positions, q_i, and velocities, $\dot{q}_i$, must be given. From q_i, the orientation, ${}^i\boldsymbol{R}_{i-1}$, and position, ${}^{i-1}\boldsymbol{p}_i$, of link i's coordinate system with respect to $i-1$'s can be determined. For revolute joints, ${}^i\boldsymbol{R}_{i-1}$ is a function of q_i and ${}^{i-1}\boldsymbol{p}_i$ is constant and can be determined off-line. For prismatic joints, ${}^i\boldsymbol{R}_{i-1}$ is constant and ${}^{i-1}\boldsymbol{p}_i$ is a function of q_i. Taken together, ${}^i\boldsymbol{R}_{i-1}$ and ${}^{i-1}\boldsymbol{p}_i$ completely specify the spatial transformation matrix, ${}^i\mathbf{X}_{i-1}$. This is used along with $\dot{q}_i$ to compute $\boldsymbol{v}_i^r$ for each link, and has been combined with the computation of the angular velocity for the sake of efficiency. The computation of ${}^i\boldsymbol{a}_{f-g}$ is accomplished using just ${}^i\boldsymbol{R}_{i-1}$, while $\boldsymbol{\zeta}_i$ depends on both ${}^i\boldsymbol{R}_{i-1}$ and ${}^{i-1}\boldsymbol{p}_i$. Finally, the computation of $\boldsymbol{\beta}_i^H$ is the same for the base and the links of the manipulator. Note that this step still includes the computation of $\boldsymbol{\beta}_i$ from the land-based algorithm which has been separated from the computation of $\boldsymbol{\beta}_i^H$ for

Table 1: AB algorithm for a URV with a manipulator.

I. Forward Kinematics Given: ${}^0\boldsymbol{R}_e$, $\boldsymbol{\omega}_0$, $\boldsymbol{v}_0$, ${}^e\boldsymbol{v}_f$, ${}^e\boldsymbol{a}_f$.

$$
\begin{aligned}
\boldsymbol{v}_0^r &= \boldsymbol{v}_0 - {}^0\boldsymbol{R}_e\,{}^e\boldsymbol{v}_f \\
{}^0\boldsymbol{a}_g &= {}^0\boldsymbol{R}_e\,{}^e\boldsymbol{a}_g \qquad \text{where } {}^e\boldsymbol{a}_g = [0 \;\; 0 \;\; g]^T \\
{}^0\boldsymbol{a}_{f-g} &= {}^0\boldsymbol{R}_e\,({}^e\boldsymbol{a}_f - {}^e\boldsymbol{a}_g) \\
\boldsymbol{\beta}_0 &= -\begin{bmatrix} \boldsymbol{\omega}_0 \times \bar{\boldsymbol{I}}_0\boldsymbol{\omega}_0 \\ \boldsymbol{\omega}_0 \times (\boldsymbol{\omega}_0 \times \boldsymbol{h}_0) \end{bmatrix} \\
\boldsymbol{\beta}_0^H &= \boldsymbol{\beta}_0 + \mathbf{I}_0^A \begin{bmatrix} \mathbf{0} \\ {}^0\boldsymbol{a}_{f-g} + \boldsymbol{\omega}_0 \times \boldsymbol{v}_0^r \end{bmatrix} - \begin{bmatrix} \tilde{\boldsymbol{\omega}}_0 & \tilde{\boldsymbol{v}}_0^r \\ \mathbf{0} & \tilde{\boldsymbol{\omega}}_0 \end{bmatrix} \mathbf{I}_0^A \begin{bmatrix} \boldsymbol{\omega}_0 \\ \boldsymbol{v}_0^r \end{bmatrix} \\
&\qquad + \mathbf{f}_0^D(\boldsymbol{\omega}_0, \boldsymbol{v}_0^r) + m_0^f \begin{bmatrix} \boldsymbol{b}_0 \times {}^0\boldsymbol{a}_{f-g} \\ {}^0\boldsymbol{a}_{f-g} \end{bmatrix}.
\end{aligned}
$$

for $i = 1, \ldots, N$. Given: ${}^i\boldsymbol{R}_{i-1}$, ${}^{i-1}\boldsymbol{p}_i$, $\dot{q}_i$.

$$
\begin{aligned}
\begin{bmatrix} \boldsymbol{\omega}_i \\ \boldsymbol{v}_i^r \end{bmatrix} &= {}^i\mathbf{X}_{i-1} \begin{bmatrix} \boldsymbol{\omega}_{i-1} \\ \boldsymbol{v}_{i-1}^r \end{bmatrix} + \boldsymbol{\phi}_i \dot{q}_i \\
{}^i\boldsymbol{a}_{f-g} &= {}^i\boldsymbol{R}_{i-1}\,{}^{i-1}\boldsymbol{a}_{f-g} \\
\boldsymbol{\zeta}_i &= \begin{bmatrix} \mathbf{0} \\ {}^i\boldsymbol{R}_{i-1}\left[\boldsymbol{\omega}_{i-1} \times (\boldsymbol{\omega}_{i-1} \times {}^{i-1}\boldsymbol{p}_i)\right] \end{bmatrix} + \begin{bmatrix} \sigma_i(\boldsymbol{\omega}_i \times \dot{q}_i\hat{\mathbf{z}}_i) \\ \bar{\sigma}_i(2\boldsymbol{\omega}_i \times \dot{q}_i\hat{\mathbf{z}}_i) \end{bmatrix} \\
\boldsymbol{\beta}_i &= -\begin{bmatrix} \boldsymbol{\omega}_i \times \bar{\boldsymbol{I}}_i\boldsymbol{\omega}_i \\ \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \boldsymbol{h}_i) \end{bmatrix} \\
\boldsymbol{\beta}_i^H &= \boldsymbol{\beta}_i + \mathbf{I}_i^A \begin{bmatrix} \mathbf{0} \\ {}^i\boldsymbol{a}_{f-g} + \boldsymbol{\omega}_i \times \boldsymbol{v}_i^r \end{bmatrix} - \begin{bmatrix} \tilde{\boldsymbol{\omega}}_i & \tilde{\boldsymbol{v}}_i^r \\ \mathbf{0} & \tilde{\boldsymbol{\omega}}_i \end{bmatrix} \mathbf{I}_i^A \begin{bmatrix} \boldsymbol{\omega}_i \\ \boldsymbol{v}_i^r \end{bmatrix} \\
&\qquad + \mathbf{f}_i^D(\boldsymbol{\omega}_i, \boldsymbol{v}_i^r) + m_i^f \begin{bmatrix} \boldsymbol{b}_i \times {}^i\boldsymbol{a}_{f-g} \\ {}^i\boldsymbol{a}_{f-g} \end{bmatrix}.
\end{aligned}
$$

end for i.

---

emphasis.

In the Backward Dynamics step, the input joint torques/forces, τ_i, must be given. Given these inputs, the backward recursion is almost identical to the land-based

Table 1: (continued.) AB algorithm for a URV with a manipulator.

II. Backward Dynamics Given: $\mathbf{I}_N^* = \mathbf{I}_N^H$, $\boldsymbol{\beta}_N^* = \boldsymbol{\beta}_N^H$ (no tip force).
for $i = N, \ldots, 1$. Given: τ_i.

$$
\begin{aligned}
\mathbf{n}_i &= \mathbf{I}_i^* \boldsymbol{\phi}_i \\
m_i^* &= \boldsymbol{\phi}_i^T \mathbf{I}_i^* \boldsymbol{\phi}_i \\
\mathbf{N}_i &= \mathbf{I}_i^* - \mathbf{n}_i (m_i^*)^{-1} \mathbf{n}_i^T \\
\tau_i^* &= \boldsymbol{\phi}_i^T \boldsymbol{\beta}_i^* + \tau_i \\
\mathbf{I}_{i-1}^* &= \mathbf{I}_{i-1}^H + {}^i\mathbf{X}_{i-1}^T \mathbf{N}_i {}^i\mathbf{X}_{i-1} \\
\boldsymbol{\beta}_{i-1}^* &= \boldsymbol{\beta}_{i-1}^H + {}^i\mathbf{X}_{i-1}^T \left[\boldsymbol{\beta}_i^* - \mathbf{N}_i \boldsymbol{\zeta}_i - \mathbf{n}_i (m_i^*)^{-1} \tau_i^* \right]
\end{aligned}
$$

end for i.

III. Forward Accelerations Given: $\mathbf{f}_0$ (thruster force).

$$
\begin{aligned}
\mathbf{a}_0' &= \left(\mathbf{I}_0^*\right)^{-1} \left(\mathbf{f}_0 + \boldsymbol{\beta}_0^*\right) \\
\mathbf{a}_0 &= \mathbf{a}_0' + \begin{bmatrix} \mathbf{0} \\ {}^0\boldsymbol{a}_g \end{bmatrix}
\end{aligned}
$$

for $i = 1, \ldots, N$. Given: ${}^i\mathbf{X}_{i-1}$.

$$
\begin{aligned}
\bar{\mathbf{a}}_i &= {}^i\mathbf{X}_{i-1} \mathbf{a}_{i-1}' + \boldsymbol{\zeta}_i \\
\ddot{q}_i &= (m_i^*)^{-1} \tau_i^* - \left[\mathbf{n}_i (m_i^*)^{-1} \right]^T \bar{\mathbf{a}}_i \\
\mathbf{a}_i' &= \bar{\mathbf{a}}_i + \boldsymbol{\phi}_i \ddot{q}_i
\end{aligned}
$$

end for i.

---

algorithm. Only the computations of $\mathbf{I}_i^*$ and $\boldsymbol{\beta}_i^*$ are affected by the hydrodynamics. They now use $\mathbf{I}_i^H$ and $\boldsymbol{\beta}_i^H$, and take into account that the matrix addition to compute $\mathbf{I}_i^H$ can be performed off-line and does not add to the amount of computation. Finally, no changes need to be made to the Forward Accelerations step.

5. Computational Requirements

In this section, the computational cost of each equation in Table 1 is determined and listed in the last two columns of Table 2 for a URV system with an manipulator containing N revolute joints. In this table, the number of floating point multiplies and divides are considered together under the label of multiplies ($\times$), and the number of additions and subtractions are combined under the label of additions ($+$). In addition, only the minimum number of trigonometric operations (one sine/cosine pair for each angle) are needed and are not included in the totals. This section also compares these results to efficient algorithms developed in our work for land-based robotic systems (without hydrodynamics) with both fixed and mobile bases.

In our implementation of the algorithm, the rotation matrix expressing the orientation of the URV body, ${}^0\boldsymbol{R}_e$, is defined by three Euler angles [30]. In the Forward Kinematics step, this matrix is used to transform quantities expressed in the earth-fixed inertial reference frame to the URV's body-fixed frame. To achieve the most efficient transformation of these quantities, this is implemented as three separate planar rotations each requiring [4M, 2A][§]. With this approach, the transformations of ${}^e\boldsymbol{v}_f$ and ${}^e\boldsymbol{a}_{f-g}$ cost [12M, 6A] each. The alternative would be to evaluate ${}^0\boldsymbol{R}_e$ ([12M, 4A]) and perform two 3×3 matrix-vector multiplies ([9M, 6A] each) for a total cost of [30M, 16A]. Since the gravitational acceleration contains only a z component, its transformation only requires [4M, 0A].

The placement of the body-fixed (0) coordinate system also has a significant effect on the amount of computation in this algorithm. To minimize the amount of computation, it is placed so that its origin and z-axis coincide with the first link's coordinate system in the manipulator. This results in a single planar rotation (through the manipulator's first joint angle about the z-axis) when transforming both Cartesian and

[§][4M, 2A] = 4 multiplies, 2 additions.

spatial quantities between these systems. Therefore, rotation of a Cartesian vector will require [4M,2A], and transformation of a spatial vector (actually planar rotations of its two component Cartesian vectors) requires [8M, 4A]. This is employed in the first iteration of the loop in the Forward Kinematics step to compute $\boldsymbol{\omega}_1$, $\boldsymbol{v}_1^r$, ${}^1\boldsymbol{a}_{f-g}$, and $\boldsymbol{\zeta}_1$.

The same concept has also been applied to the transformations between adjacent links in the serial chain. With the use of MDH parameters [19, 20], transformation of Cartesian vectors can be accomplished with two planar rotations (about the x and z axes) for a cost of [8M, 4A]. Featherstone [17] also applied this concept to spatial transformations which are treated as two separate planar screws. Each planar screw transformation of a spatial vector costs [10M, 6A] for a total of [20M, 12A] between adjacent coordinate systems [20]. This accounts for the computational requirements of $\boldsymbol{\omega}_i$, $\boldsymbol{v}_i^r$, ${}^i\boldsymbol{a}_{f-g}$, and $\boldsymbol{\zeta}_i$ for $i > 1$.

Computation of the hydrodynamic bias force, $\boldsymbol{\beta}_i^H$, is the most significant in the Forward Kinematics step. First, the computation of $\boldsymbol{\beta}_i$ is accomplished with a Cartesian matrix-vector multiply and three cross products totalling [27M, 15A]. Then, the next two terms of $\boldsymbol{\beta}_i^H$ compute the added mass bias force, $\boldsymbol{\beta}_i^A$. The first is computed with a cross product, Cartesian vector addition, two Cartesian matrix-vector multiplies, and a spatial vector addition for a total of [24M, 24A]. The second term is obtained with a spatial matrix-vector multiply, three cross products, a Cartesian vector addition and a spatial vector addition for a total of [54M, 48A].

Computation of the drag force, $\mathbf{f}_i^D$ is accomplished using Eqs. (20), (21), and (22). The x-component of the drag force in Eq. (22) requires [2M, 0A] for the on-line part of the computation. The integrals in the first two equations, can be numerically determined using the Gauss-quadrature method which replaces the integral with a weighted sum of the term within the integral evaluated at four points over its range. The terms to be summed can be computed with [9M, 3A] and a square root (counted

as a multiplication) which includes the multiplication by the weight and the computation of $\boldsymbol{v}^r(d)$ from Eq. (17). This is performed four times requiring [40M, 12A]. Then, the results are added together ([0M, 12A]) noting that both vectors contain one zero element, and then multiplied by the constant outside of the integral ([4M, 0A]). Therefore, computing $\mathbf{f}_i^D$ with five non-zero terms and adding it to the bias force requires a total of [46M, 29A].

Finally, the total buoyancy force, the last term in the computation of $\boldsymbol{\beta}_i^H$, requires [9M, 9A] noting that the acceleration, ${}^i\boldsymbol{a}_{f-g}$, should be multiplied by the mass of the displaced fluid before the cross product is performed. Therefore, the computation of $\boldsymbol{\beta}_i^H$ requires [133M, 110A]. This also applies to the computation of $\boldsymbol{\beta}_0^H$ for the vehicle body which implies that the vehicle body has also been approximated by a cylinder for the drag computation. If desired, equivalent methods can be developed for rectangular solids which require comparable amounts of computation.

A number of steps can be taken to reduce the amount of computation in the Backward Dynamics recursion. First, $\mathbf{n}_i$, m_i^*, and $\mathbf{N}_i$ are constant for $i = N$ and can be computed off-line along with $(m_i^*)^{-1}$ and $\mathbf{n}_i(m_i^*)^{-1}$. For the other iterations, the computation of $\mathbf{n}_i$ reduces to selecting the third column of $\mathbf{I}_i^*$, $(m_i^*)^{-1}$ simplifies to a scalar reciprocal, and the third element of $\mathbf{n}_i(m_i^*)^{-1}$ is one. The matrix, $\mathbf{N}_i$, is the ith AB inertia after it has been reflected across joint i. Because the joint is free to move, it contains a zero row and column as follows:

$$\mathbf{N}_i = \begin{bmatrix} N_{11} & N_{12} & 0 & N_{14} & N_{15} & N_{16} \\ N_{12} & N_{22} & 0 & N_{24} & N_{25} & N_{26} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ N_{14} & N_{24} & 0 & N_{44} & N_{45} & N_{46} \\ N_{15} & N_{25} & 0 & N_{45} & N_{55} & N_{56} \\ N_{16} & N_{26} & 0 & N_{46} & N_{56} & N_{66} \end{bmatrix}. \tag{32}$$

This reflected inertia matrix is also symmetric (notice the indices), and only 15 of its elements must be computed, which requires [1M, 1A] each. The most costly

step in this recursion is the spatial congruence transformation of this matrix in the computation of the AB inertia, $\mathbf{I}^*_{i-1}$. A new method developed in [12, 31] requires only [70M, 71A] and is accomplished by breaking this computation into two congruence transformations involving planar screws analogous to the decomposition of spatial transformations above. To complete the computation of $\mathbf{I}^*_{i-1}$ requires the addition of two symmetric 6×6 matrices, requiring 21 additions, for a total of [70M, 92A]. Due to the placement of the body (0) coordinate system, the last iteration of this computation ($\mathbf{I}^*_0$) can be performed using a single planar rotation matrix (simpler than a planar screw) which results in [22M, 40A] for this step.

In the Forward Accelerations step, the biased base acceleration, $\mathbf{a}'_0$, is most efficiently computed using Cholesky decomposition since the AB inertia is symmetric and positive definite. Note that this step does not compute the inverse of this matrix, but only the solution to the matrix equation which requires only [86M, 65A] for the 6×6 system [10]. If present, a thruster force on the vehicle body, $\mathbf{f}_0$, must first be added for a total of [86M, 71A]. The true acceleration of the vehicle body is obtained by adding the gravitational acceleration to the result ([0M,3A]). Finally, the amount of computation in the forward recursion to compute the joint accelerations is straightforward. However, note that the third element of $\boldsymbol{\zeta}_i$ is zero and the computation of $\mathbf{a}'_N$ is not required.

This particularly efficient implementation of the URV algorithm is based on versions that were developed for simulation of land-based manipulator systems with and without a mobile base [12], and the results of which are also listed in Table 2. The requirements for a manipulator with a fixed base are listed in the first columns of the table and results in $[(224N - 259)\text{M}, (205N - 248)\text{A}]$ for a system with N revolute joints. This result omits the computation of hydrodynamics and the effects of a mobile base, and it makes use of the fact that the base has zero acceleration and velocity to reduce the amount of computation of the quantities for the first few

Table 2: Computational requirements of the AB simulation algorithm.

	Fixed Base		Mobile Base		w/ Hydrodynamics	
	$\times$	$+$	$\times$	$+$	$\times$	$+$
$\boldsymbol{v}_0^r$	—	—	—	—	12	9
${}^0\boldsymbol{a}_g$	—	—	4	—	4	—
${}^0\boldsymbol{a}_{f-g}$	—	—	—	—	12	7
$\boldsymbol{\beta}_0$	—	—	27	15	27	15
$\boldsymbol{\beta}_0^H$	—	—	—	—	133	110
$\boldsymbol{\omega}_i$	$8N-12$	$5N-9$	$8N-4$	$5N-2$	—	—
$\boldsymbol{\omega}_i$, $\boldsymbol{v}_i^r$	—	—	—	—	$20N-12$	$13N-8$
${}^i\boldsymbol{a}_{f-g}$	—	—	—	—	$8N-4$	$4N-2$
$\boldsymbol{\zeta}_i$	$22N-33$	$10N-18$	$22N-20$	$10N-10$	$22N-20$	$10N-10$
$\boldsymbol{\beta}_i$	$27N-19$	$15N-15$	$27N$	$15N$	$27N$	$15N$
$\boldsymbol{\beta}_i^H$	—	—	—	—	$133N$	$110N$
$\mathbf{n}_i$	—	—	—	—	—	—
$(m_i^*)^{-1}$	$N-1$	—	$N-1$	—	$N-1$	—
$\mathbf{n}_i\,(m_i^*)^{-1}$	$5N-5$	—	$5N-5$	—	$5N-5$	—
$\mathbf{N}_i$	$15N-30$	$15N-30$	$15N-15$	$15N-15$	$15N-15$	$15N-15$
τ_i^*	—	N	—	N	—	N
$\mathbf{I}_{i-1}^*$	$70N-80$	$86N-94$	$70N-58$	$86N-59$	$70N-48$	$92N-52$
$\boldsymbol{\beta}_{i-1}^*$	$50N-54$	$49N-55$	$50N-31$	$49N-27$	$50N-27$	$49N-23$
$\mathbf{a}_0'$	—	—	86	71	86	71
$\mathbf{a}_0$	—	—	—	3	—	3
$\bar{\mathbf{a}}_i$	$20N-22$	$17N-21$	$20N-12$	$17N-11$	$20N-12$	$17N-11$
$\ddot{q}_i$	$6N-3$	$6N-4$	$6N-1$	$6N-1$	$6N$	$6N$
$\mathbf{a}_i'$	—	$N-2$	—	$N-1$	—	$N-1$
Total:	$224N-259$	$205N-248$	$224N-30$	$205N-37$	$377N+130$	$333N+93$
$(N=6)$	(1085)	(982)	(1314)	(1193)	(2392)	(2091)

links of the manipulator. This can be directly compared to the best previous result, $[(250N-222)\text{M}, (220N-248)\text{A}]$, reported by Brandl, Johanni, and Otter [22]. When $N = 6$, the result reported here requires nearly 15% fewer operations to compute the dynamics.

With the addition of a mobile base, the algorithm could be used to simulate space-based robotic systems (with rigid links). The resulting computational requirements are listed in the middle of Table 2. This involves the addition of another rigid body and six more DOFs to the system, but only requires an additional [229M, 211A] for a total of $[(224N - 30)\text{M}, (205N - 37)\text{A}]$. This is almost equivalent to the cost of adding another link with a revolute joint to the manipulator.

The primary difference in computation between this mobile base algorithm and the URV algorithm occurs in the Forward Kinematics step where additional velocities, accelerations, and bias forces in $\boldsymbol{\beta}_i^H$ are computed for hydrodynamic effects. The total number of floating point operations for this system is $[(377N+130)\text{M}, (333N+93)\text{A}]$. With $N = 6$ (all revolute joints), this corresponds to the cost needed to compute the dynamics and hydrodynamics for the Monterey Bay Aquarium Research Institute's URV with the 6 DOF Schilling Titan II manipulator. This represents a 75% increase in computation over the mobile base simulation (without hydrodynamics), and approximately a 115% increase in computation over the simulation of this manipulator with a fixed base.

6. DynaMechs: An Object-Oriented Simulator

This paper provides all of the major theory needed to produce a dynamic simulator that efficiently incorporates hydrodynamic effects for underwater vehicle systems. The algorithm presented thus far has been slightly modified so that it is capable of

computing the dynamics for a large class of tree-structured mechanisms having star topologies. This class includes robotic systems with fixed or mobile bases, with and without hydrodynamics, and with multiple chains. An efficient implementation of this general algorithm has been achieved through the use of object-oriented design (OOD) techniques in C++ as part of a larger project to develop a real-time graphical simulation system for URVs as described in [15]. The result is an integrated simulation software package, called **DynaMechs**[¶], primarily for terrestrial and underwater robotic systems. It covers industrial robots, multilegged vehicles [32], URVs with any number of manipulators, and can even simulate space-based robotic systems without flexible links.

The completed graphical simulation system also includes a Spaceball (a six-axis force/torque sensor) for user input of thruster, manipulator, and viewpoint control commands, and 3D graphical display of the system and its environment to visualize the output of the simulation. A scene from this display is shown in Figure 8 which was developed using SGI's 3D modeling package called Inventor and executed on a Silicon Graphics (SGI) Indigo^2 Extreme workstation with a 150MHz MIPS R4400 processor. The runtime performance of the simulation algorithm (excluding the overhead for the graphical display) for the three configurations listed in Table 2 was also measured. Without hydrodynamics, computation of the Schilling manipulator's dynamics requires an average of 0.30 and 0.38 ms for fixed and mobile base systems, respectively. The ROV/manipulator system with hydrodynamics requires an average of 0.52 ms. This is more than adequate for real-time performance during normal operation of the ROV system without "hard" contacts with the environment.

[¶]Pronounced "dynamics," and stands for Dynamics of Mechanisms. This software is available via anonymous FTP from `taurus.cs.nps.navy.mil` in the `/pub/dynamechs/` directory.

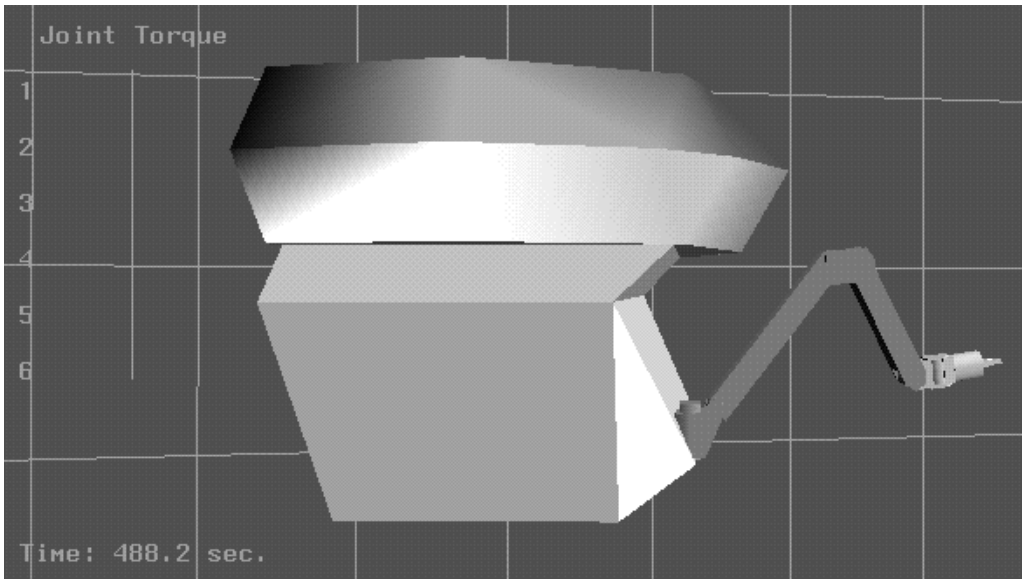


Figure 8: Scene from a **DynaMechs** simulation of the *Tiburon/Schilling* system.

7. Summary and Conclusions

In this paper, an efficient algorithm has been developed for dynamic simulation of an underwater vehicle equipped with a manipulator. Since many efficient algorithms for the simulation of land-based manipulators have been developed in the past, one was chosen as the basis for our work. The system for which this work has been developed is the Monterey Bay Aquarium Research Institute's ROV, *Tiburon*, with a Shilling manipulator. Based on the number of DOFs for this manipulator, the O(N) AB simulation algorithm is the most efficient. A review of this algorithm for systems with fixed bases from [11] has been presented. An especially efficient version of this algorithm was developed in [12] which requires [$(224N - 259)$M, $(205N - 248)$A] for systems with N revolute joints which represents an improvement of 15% over the results in [22]. With the modifications made to include the effects of a mobile base the algorithm requires [$(224N - 30)$M, $(205N - 37)$A].

The next goal of this work was to efficiently incorporate hydrodynamic effects

into this algorithm. A number of hydrodynamic effects on a single rigid body were identified from previous work in [13, 14] which include added mass, drag, fluid acceleration and buoyancy forces. Equations to compute these effects are derived here in a form consistent with traditional robot dynamics algorithms. Then, this result is extended to systems with serial chains of rigid bodies and efficiently incorporated into the AB algorithm. A detailed analysis of the amount of computation required by the resulting algorithm also has been carried out, and shown to require $[(377N + 130)\text{M}, (333N + 93)\text{A}]$ for a URV with a manipulator that has N revolute joints. For the MBARI ROV with the Schilling Arm ($N = 6$), the computation of the dynamics requires [2392M, 2091A]. An implementation of the algorithm in the **DynaMechs** software package can compute the dynamics of this URV system in an average of 0.52 ms on an SGI workstation with a MIPS R4400 processor, which is fast enough to achieve real-time simulation rates.

The purpose of this work has been to provide an integrated computational framework, with multibody dynamics and hydrodynamics included, for efficient simulation of a variety of land-based and underwater robotic systems. Now that we have provided the foundational work, others can apply this to their system, after deriving parameters for the model, to determine the important hydrodynamic effects. With an actual URV system, experimental results can be obtained and the accuracy of the models for each of the hydrodynamic forces can be measured within the context of a specific application to see if the models are sufficient. There may be cases in which the computation of the hydrodynamic forces needs to be altered to provide a more complex or perhaps simpler model. The advantage of this computational framework is that it can accommodate a variety of models in which the hydrodynamic forces on a link are computed as a function of its velocity and acceleration and that of the fluid without significantly affecting its basic efficiency. Hopefully, this work will contribute to the growing application of URV systems in underwater environments.

8. Acknowledgments

The authors would like to thank Dr. Anthony Healey of the Naval Postgraduate School for informative discussions on hydrodynamics, and James B. Newman of MBARI for access to and involvement in the *Tiburon* project.

A. Derivation of the Added Mass Force Equation

In this appendix, the added mass force equation, Eq. (11), is derived in spatial notation beginning with Newman's equations for this force from [24]. Before we begin, some of his notation should be defined. First, the ijth element of the 6×6 added mass matrix is denoted by m_{ij}. The translational velocity expressed in the body-fixed coordinate system is given by the following Cartesian vector:

$$\boldsymbol{v}_b = \begin{bmatrix} U_1 & U_2 & U_3 \end{bmatrix}^T, \tag{33}$$

and a redundant notation is used to define the elements of angular velocity:

$$\boldsymbol{\omega}_b = \begin{bmatrix} U_4 & U_5 & U_6 \end{bmatrix}^T = \begin{bmatrix} \Omega_1 & \Omega_2 & \Omega_3 \end{bmatrix}^T. \tag{34}$$

Finally an "indicial notation" for the cross product operation is defined. The jth component of the result is given as follows:

$$(\boldsymbol{b} \times \boldsymbol{a})_j = \epsilon_{jkl}\, b_k\, a_l, \tag{35}$$

where summations are implied on the right hand side for both k and l from 1 to 3.

The equation to compute the three components of translational force from [24] is given by [Eq. (115), Ch. 4]:

$$F_j = -\dot{U}_i m_{ji} - \epsilon_{jkl} U_i \Omega_k m_{li}, \qquad j = 1, 2, 3, \tag{36}$$

where, again, summations are implied for k and l from 1 to 3, and for i from 1 to 6. To obtain a vector equation, the summation on i is made explicit and the cross product term is isolated as follows:

$$F_j = -\sum_{i=1}^{6} m_{ji}\dot{U}_i - \sum_{i=1}^{6} U_i \left(\epsilon_{jkl}\Omega_k m_{li}\right), \qquad j = 1, 2, 3. \tag{37}$$

Then, the summations are expanded and the cross products are expressed in vector notation:

$$\begin{aligned} F_j = &-\left(m_{j1}\dot{U}_1 + m_{j2}\dot{U}_2 + \cdots + m_{j6}\dot{U}_6\right) - \left[U_1 \left(\boldsymbol{\omega}_b \times \begin{bmatrix} m_{11} \\ m_{21} \\ m_{31} \end{bmatrix}\right)_j + \right. \\ &\left. U_2 \left(\boldsymbol{\omega}_b \times \begin{bmatrix} m_{12} \\ m_{22} \\ m_{32} \end{bmatrix}\right)_j + \cdots + U_6 \left(\boldsymbol{\omega}_b \times \begin{bmatrix} m_{16} \\ m_{26} \\ m_{36} \end{bmatrix}\right)_j\right]. \end{aligned} \tag{38}$$

Finally, this is written in a form consistent with our spatial notation as follows:

$$\boldsymbol{f}_b^A = -\left[\mathbf{M}_{12} \;\; \mathbf{M}_{11}\right] \begin{bmatrix} \dot{\boldsymbol{\omega}}_b \\ \dot{\boldsymbol{v}}_b \end{bmatrix} - \tilde{\boldsymbol{\omega}}_b \left[\mathbf{M}_{12} \;\; \mathbf{M}_{11}\right] \begin{bmatrix} \boldsymbol{\omega}_b \\ \boldsymbol{v}_b \end{bmatrix}, \tag{39}$$

where $\boldsymbol{f}_b^A = [F_1 \; F_2 \; F_3]^T$ and the 3×3 blocks of added mass components are defined as follows:

$$\left[\mathbf{M}_{12} \;\; \mathbf{M}_{11}\right] = \left[\begin{array}{ccc|ccc} m_{14} & m_{15} & m_{16} & m_{11} & m_{12} & m_{13} \\ m_{24} & m_{25} & m_{26} & m_{21} & m_{22} & m_{23} \\ m_{34} & m_{35} & m_{36} & m_{31} & m_{32} & m_{33} \end{array}\right]. \tag{40}$$

Note that these blocks are "swapped" because spatial notation requires that angular components of spatial vectors appear above the translational components which is the opposite of the convention used in [24].

The equations for the moment exerted on the body due to added mass is given in [24] as follows [Eq. (116), Ch. 4]:

$$N_j = -\dot{U}_i m_{j+3,i} - \epsilon_{jkl} U_i \Omega_k m_{l+3,i} - \epsilon_{jkl} U_i U_k m_{li}, \qquad j = 1, 2, 3, \tag{41}$$

where the components of moment have been specified with N_j instead of the M_j used in [24] to avoid confusion with the added mass coefficients, and the implicit summations of Eq. (36) apply here as well. This can be rewritten, as before, to highlight the summation over i and the cross product operation, as follows:

$$N_j = -\sum_{i=1}^{6} m_{j+3,i}\dot{U}_i - \sum_{i=1}^{6} U_i \left(\epsilon_{jkl}\Omega_k m_{l+3,i}\right) - \sum_{i=1}^{6} U_i \left(\epsilon_{jkl} U_k m_{li}\right). \tag{42}$$

In Cartesian vector notation, this is written as follows:

$$\begin{aligned} N_j = &-\left(m_{j+3,1}\dot{U}_1 + m_{j+3,2}\dot{U}_2 + \cdots + m_{j+3,6}\dot{U}_6\right) - \\ &\left[U_1 \left(\boldsymbol{\omega}_b \times \begin{bmatrix} m_{41} \\ m_{51} \\ m_{61} \end{bmatrix}\right)_j + U_2 \left(\boldsymbol{\omega}_b \times \begin{bmatrix} m_{42} \\ m_{52} \\ m_{62} \end{bmatrix}\right)_j + \cdots + U_6 \left(\boldsymbol{\omega}_b \times \begin{bmatrix} m_{46} \\ m_{56} \\ m_{66} \end{bmatrix}\right)_j\right] - \\ &\left[U_1 \left(\boldsymbol{v}_b \times \begin{bmatrix} m_{11} \\ m_{21} \\ m_{31} \end{bmatrix}\right)_j + U_2 \left(\boldsymbol{v}_b \times \begin{bmatrix} m_{12} \\ m_{22} \\ m_{32} \end{bmatrix}\right)_j + \cdots + U_6 \left(\boldsymbol{v}_b \times \begin{bmatrix} m_{16} \\ m_{26} \\ m_{36} \end{bmatrix}\right)_j\right], \end{aligned} \tag{43}$$

for $j = 1, 2, 3$. In spatial notation, this reduces to:

$$\boldsymbol{n}_b^A = -\left[\mathbf{M}_{22} \;\; \mathbf{M}_{21}\right] \begin{bmatrix} \dot{\boldsymbol{\omega}}_b \\ \dot{\boldsymbol{v}}_b \end{bmatrix} - \tilde{\boldsymbol{\omega}}_b \left[\mathbf{M}_{22} \;\; \mathbf{M}_{21}\right] \begin{bmatrix} \boldsymbol{\omega}_b \\ \boldsymbol{v}_b \end{bmatrix} - \tilde{\boldsymbol{v}}_b \left[\mathbf{M}_{12} \;\; \mathbf{M}_{11}\right] \begin{bmatrix} \boldsymbol{\omega}_b \\ \boldsymbol{v}_b \end{bmatrix}, \tag{44}$$

where $\boldsymbol{n}_A = [N_1 \; N_2 \; N_3]^T$ and

$$\left[\mathbf{M}_{22} \;\; \mathbf{M}_{21}\right] = \left[\begin{array}{ccc|ccc} m_{44} & m_{45} & m_{46} & m_{41} & m_{42} & m_{63} \\ m_{54} & m_{55} & m_{56} & m_{51} & m_{52} & m_{53} \\ m_{64} & m_{65} & m_{66} & m_{61} & m_{62} & m_{43} \end{array}\right]. \tag{45}$$

By combining Eqs. (44) and (39), the desired added mass spatial force equation is obtained as:

$$\mathbf{f}_b^A = \begin{bmatrix} \boldsymbol{n}_b^A \\ \boldsymbol{f}_b^A \end{bmatrix} = -\mathbf{I}_b^A \begin{bmatrix} \dot{\boldsymbol{\omega}}_b \\ \dot{\boldsymbol{v}}_b \end{bmatrix} - \begin{bmatrix} \tilde{\boldsymbol{\omega}}_b & \tilde{\boldsymbol{v}}_b \\ \mathbf{0} & \tilde{\boldsymbol{\omega}}_b \end{bmatrix} \mathbf{I}_b^A \begin{bmatrix} \boldsymbol{\omega}_b \\ \boldsymbol{v}_b \end{bmatrix}, \tag{46}$$

where our definition for the added mass matrix, $\mathbf{I}_A$, is written in terms of Newman's coefficients as:

$$\mathbf{I}_b^A = \begin{bmatrix} \mathbf{M}_{22} & \mathbf{M}_{21} \\ \mathbf{M}_{12} & \mathbf{M}_{11} \end{bmatrix}. \tag{47}$$

References

[1] J. Yuh, "Special Issue on Underwater Robotics," *The Journal of Robotics Systems*, vol. 8, pp. 291–412, June 1991.

[2] *Proc. of the 20th Annual Technical Symposium and Exhibition of the Association for Unmanned Vehicle Systems*, (Washington, D.C.), Association for Unmanned Vehicle Systems, June 1993.

[3] T. I. Fossen, *Guidance and Control of Ocean Vehicles*. New York: John Wiley & Sons Inc., 1994.

[4] M. J. Zyda, R. B. McGhee, S. Kwak, D. B. Nordman, R. C. Rogers, and D. Marco, "Three-Dimensional Visualization of Mission Planning and Control for the NPS Autonomous Underwater Vehicle," *IEEE Journal of Oceanic Engineering*, vol. 15, no. 3, pp. 217–221, 1990.

[5] D. P. Brutzman, Y. Kanayama, and M. J. Zyda, "Integrated Simulation for Rapid Development of Autonomous Underwater Vehicles," in *Proc. of IEEE Symposium on Autonomous Underwater Vehicle Technology*, (Washington, D.C.), pp. 3–10, June 1992.

[6] L. Conway, R. Volz, and M. Walker, "Tele-Autonomous Systems: Methods and Architectures for Intermingling Autonomous and Telerobotic Technology," in *Proc. of 1987 IEEE Int. Conf. on Robotics and Automation*, (Raleigh, NC), pp. 1121–1130, 1987.

[7] J. Funda and R. P. Paul, "Teleprogramming: Overcoming Communication Delays in Remote Manipulation," in *Proc. of 1st IARP Workshop on Mobile Robots for Subsea Environments*, (Monterey, CA), pp. 155–162, October 1990.

[8] J. B. Newman and B. H. Robison, "Development of a Dedicated ROV for Ocean Science," *Marine Technology Society Journal*, vol. 26, pp. 46–53, Winter 1992.

[9] Anon., "Titan II High Resolution Bilateral Force Feedback Remote Manipulator System," Technical Report, Schilling Development, Inc., Davis, CA, April 1991.

[10] M. W. Walker and D. E. Orin, "Efficient Dynamic Computer Simulation of Robotic Mechanisms," *Journal of Dynamic Systems, Measurement, and Control*, vol. 104, pp. 205–211, September 1982.

[11] R. Featherstone, "The Calculation of Robot Dynamics Using Articulated-Body Inertias," *The Int. Journal of Robotics Research,* The MIT Press, vol. 2, pp. 13–30, Spring 1983.

[12] S. McMillan, *Computational Dynamics for Robotic Systems on Land and Under Water.* PhD dissertation, The Ohio State University, Columbus, Ohio, Summer 1994.

[13] J. Yuh, "Modeling and Control of Underwater Robotic Vehicles," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, pp. 1475–1483, November/December 1990.

[14] K. Ioi and K. Itoh, "Modelling and Simulation of an Underwater Manipulator," *Advanced Robotics*, vol. 4, no. 4, pp. 303–317, 1990.

[15] S. McMillan, D. E. Orin, and R. B. McGhee, "DynaMechs: An Object Oriented Software Package for Efficient Dynamic Simulation of Underwater Robotic Vehicles," in *Underwater Robotic Vehicles: Design and Control* (J. Yuh, ed.), pp. 73–98, Albuquerque, NM: TSI Press, 1995.

[16] R. E. Roberson and R. Schwertassek, *Dynamics of Multibody Systems.* Berlin: Springer-Verlag, 1988.

[17] R. Featherstone, *Robot Dynamics Algorithms.* Boston: Kluwer Academic Publishers, 1987.

[18] K. W. Lilly, *Efficient Dynamic Simulation of Robotic Mechanisms.* Norwell, MA: Kluwer Academic Publishers, 1993.

[19] J. J. Craig, *Introduction to Robotics: Mechanics and Control.* Reading, MA: Addison-Wesley, 1986.

[20] K. W. Lilly and D. E. Orin, "Alternate Formulations for the Manipulator Inertia Matrix," *International Journal of Robotics Research,* The MIT Press, vol. 10, pp. 64–74, February 1991.

[21] G. Dahlquist and Å. Björck, *Numerical Methods.* Englewood Cliffs, NJ: Prentice Hall, 1974.

[22] H. Brandl, R. Johanni, and M. Otter, "A Very Efficient Algorithm for the Simulation of Robots and Similar Multibody Systems Without Inversion of the Mass Matrix," in *Proc. of IFAC/IFIP/IMACS Int. Symp. on Theory of Robots*, (Vienna, Austria), December 1986.

[23] I. H. Shames, *Mechanics of Fluids*. New York, NY: McGraw-Hill, 1962.

[24] J. N. Newman, *Marine Hydrodynamics*. Cambridge, MA: The MIT Press, 1977.

[25] "Controllability," in *Principles of Naval Architecture* (E. Lewis, ed.), vol. 3, (New York, NY), Society of Naval Architects and Marine Engineers, 1987.

[26] J. L. Synge and B. A. Griffith, *Principles of Mechanics*. New York, NY: McGraw Hill, 1949.

[27] T. Sarpkaya and M. Isaacson, *Mechanics of Wave Forces on Offshore Structures*. New York, NY: Van Nostrand Reinhold Co., 1981.

[28] S. K. Chakrabarti, W. A. Tam, and A. L. Wolbert, "Wave Forces on a Randomly Oriented Tube," in *Proc. of the O.T.C.*, no. 2190, (Houston, TX), 1975.

[29] O. M. Griffin and S. E. Ramberg, "Some Recent Studies of Vortex Shedding with Application to Marine Tubulars and Risers," *Transactions of the ASME*, vol. 104, pp. 2–13, March 1982.

[30] J. M. Cooke, M. J. Zyda, D. R. Pratt, and R. B. McGhee, "NPSNET: Flight Simulation Dynamic Modeling Using Quaternions," *Presence*, vol. 1, pp. 404–420, Fall 1992.

[31] S. McMillan and D. E. Orin, "Efficient Computation of Articulated-Body Inertias Using Successive Axial Screws," *IEEE Trans. on Robotics and Automation*, vol. 11, pp. 606–611, August 1995.

[32] K. Yoneda, K. Suzuki, and Y. Kanayama, "Gait Planning for Versatile Motion of a Six Legged Robot," in *IEEE Int. Conf. on Robotics and Automation*, (San Diego, CA), pp. 1338–1343, 1994.