CHAPTER VII

Efficient Simulation of Underwater Robotic Vehicle Systems

7.1 Introduction

The primary goal of this chapter is to develop a general, but efficient algorithm to compute the dynamics of underwater robotic vehicle (URV) systems with multiple chains. The remotely operated vehicle (ROV), *Tiburon*, with a single Schilling manipulator and the six-legged Aquarobot, as illustrated in Figures 18 and 19, are specifically targeted in this development. The Aquarobot has the same general topology as legged vehicles examined in Chapter 5, so the decoupled tree-structure (DTS) approach is a general, but efficient, approach for modeling this system. This approach can also be applied to the ROV/manipulator system in which the reference member has a single chain attached.

Using DTS modeling, the resulting systems require an open chain dynamics algorithm, and in Chapter 5, Walker and Orin's Composite Rigid Body (CRB) method [14] was used in the development for such systems on land (or in space). Compared to results obtained later in this chapter for an equivalent algorithm based on Featherstone's Articulated Body (AB) method [15], the CRB method is more efficient only when the number of degrees of freedom in each chain is less than four. Disregarding the additional computation for hydrodynamic effects, the CRB method is more efficient for the Aquarobot where $N = 3$, but the AB algorithm is more efficient for the ROV/manipulator system because the number of links in the chain is greater than this crossover point.

This conclusion is correct under the assumption that the incorporation of the hydrodynamic effects adds the same amount of computation to both types of algorithms. This, however, is not the case. While, the computation of the added mass bias, drag, and total buoyancy forces does add the same amount to both algorithms, the inclusion of the added mass matrix has a very different effect on both. As will be shown later in this chapter, this matrix will be added to spatial inertias of each rigid body in the system and the special structure of the latter [shown in Eq. (1.7)]

is replaced by a full symmetric matrix. In the AB algorithm, this matrix is added to AB inertia matrices which are already full symmetric matrices and the procedure has no effect on the computational requirements of subsequent operations. In the CRB algorithm, subsequent operations using this matrix depended on the special structure in Eq. (1.7) to achieve greater efficiency and compete with the AB algorithms. Transforms of the new matrix (with the added mass) require nearly twice as many operations and the resulting CRB algorithm for URV systems will always be less efficient than the AB algorithm.

Therefore, the AB method is used as the basis for the efficient URV dynamics algorithm developed in this chapter which uses the procedures for computing hydrodynamic forces developed in the previous chapter. In the next section, a derivation of the AB dynamic equations to model systems with mobile bases and multiple serial chains is presented using the spatial notation in this dissertation. In Section 7.3, an efficient procedure for incorporating the hydrodynamics terms of Chapter 6 is presented. Then, the complete AB/DTS algorithm for general multiple chain URV systems is presented along with its computational requirements. This will be used in Chapter 8, as the basis for an efficient implementation of the dynamics for general robotic systems using an object oriented language.

This algorithm is capable of simulating both the Aquarobot and the ROV with a single manipulator. However, like the latter, many URV systems are characterized by a single serial chain. In this case, the general algorithm is not necessarily the most efficient approach. In order to obtain more efficient algorithms for these systems, two modifications to the general algorithm are examined in Section 7.5. The first is a more efficient kinematic model specifically for single chains. With additional assumptions, reasonable simplifications to the dynamic model can also be made which result in further reductions in the amount of computation. Finally, the results using these two approaches are compared to the general simulation algorithm.

7.2 Articulated Body Dynamics

In this section, the equations used in the AB method for computing robot dynamics are derived using the spatial notation presented in this dissertation. This discussion begins with an overview of the equations describing the kinematics and dynamics of one rigid body (link) within a serial chain. These are used to derive the principal equations in the algorithm to compute the inertias of articulated bodies and the forces exerted on them. Then, additional equations are presented to model more complex systems having mobile bases with multiple serial chains attached. Finally, the efficient procedure for including gravitational effects is presented.

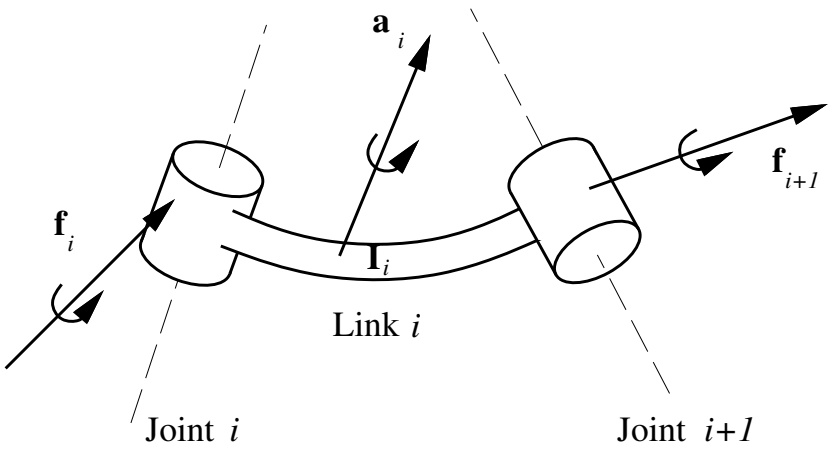

Figure 24: Rigid body dynamics for a link.

7.2.1 The Basics

The derivation of the AB algorithm begins with two basic equations for each link in the chain. The first is a kinematic equation to compute the spatial acceleration of each link i, $\mathbf{a}_i$. This is a function of the acceleration of the inboard link, $\mathbf{a}_{i-1}$, and the relative acceleration between the two links or the joint acceleration, $\ddot{q}_i$:

$$\mathbf{a}_i = {}^i\mathbf{X}_{i-1}\,\mathbf{a}_{i-1} + \boldsymbol{\phi}_i\ddot{q}_i + \boldsymbol{\zeta}_i, \tag{7.1}$$

where $\boldsymbol{\zeta}_i$ is the vector of coriolis and centripetal accelerations that are a function of known link velocities. For the single degree of freedom revolute and prismatic joints, this term is computed as follows:

$$\boldsymbol{\zeta}_i = \begin{bmatrix} \mathbf{0} \\ {}^i\boldsymbol{R}_{i-1}\left[\boldsymbol{\omega}_{i-1} \times (\boldsymbol{\omega}_{i-1} \times {}^{i-1}\boldsymbol{p}_i)\right] \end{bmatrix} + \begin{bmatrix} \sigma_i(\boldsymbol{\omega}_i \times \dot{q}_i\hat{\mathbf{z}}_i) \\ \bar{\sigma}_i(2\boldsymbol{\omega}_i \times \dot{q}_i\hat{\mathbf{z}}_i) \end{bmatrix}, \tag{7.2}$$

where $\sigma_i = 1$ if joint i is revolute, and $\sigma_i = 0$ if the joint is prismatic.

The second equation computes the dynamics for each link. As illustrated in Figure 24, this force balance on link i, ignoring gravity, is given as follows:

$$\mathbf{f}_i - {}^{i+1}\mathbf{X}_i^T\mathbf{f}_{i+1} = \mathbf{I}_i\mathbf{a}_i - \boldsymbol{\beta}_i, \tag{7.3}$$

where

$$\boldsymbol{\beta}_i = -\begin{bmatrix} \boldsymbol{\omega}_i \times \bar{\boldsymbol{I}}_i\boldsymbol{\omega}_i \\ \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \boldsymbol{h}_i) \end{bmatrix} \tag{7.4}$$

is the vector of bias forces that are a function of link velocities. The vector, $\mathbf{f}_i$, is the spatial force exerted on link i by its inboard neighbor and contains the effect of the joint force/torque, τ_i, as follows:

$$\tau_i = \boldsymbol{\phi}_i^T \mathbf{f}_i, \tag{7.5}$$

which is given in the simulation problem. Likewise, $\mathbf{f}_{i+1}$ is the force exerted by link i on the outboard link and contains the effect of τ_{i+1}.

Eq. (7.1) implies that a recursion from the base of the chain to the end-effector may be used to compute the link spatial accelerations when the acceleration of the base, $\mathbf{a}_0$, is known (it is zero for a fixed base), and the joint accelerations, $\ddot{q}_i$, are known. Eq. (7.3) implies a recursion from the end-effector to the base to determine all of the link forces once the link accelerations have been determined. These two sets of equations define the outward and inward recursions associated with the *inverse* dynamics algorithm which computes the joint torques to achieve a given motion, and is presented in Appendix B.

Taken together, Eqs. (7.1) and (7.3) also define the set of equations that need to be solved for the *forward* dynamics problem in which the joint state (position and velocity) and inputs (force/torque) are known and the joint accelerations are computed. For a serial chain with N links, they define the $12N$ equations where the $12N$ unknowns consist of the $6N$ elements of the spatial acceleration vectors, $\mathbf{a}_i$, the $5N$ unknown elements of the spatial force vectors, $\mathbf{f}_i$ (the sixth element of each vector is the known joint torque/force), and the N joint accelerations, $\ddot{q}_i$. This implies that a $12N \times 12N$ system of equations can be constructed and solved simultaneously for the unknown quantities. Featherstone has derived a much more efficient approach which uses the concept of Articulated Body (AB) inertias [15].

7.2.2 Articulated Body Inertias and Forces

The most important step in the AB algorithm involves the computation of AB inertias. Instead of using the force balance equation for a single link as in Eq. (7.3), an expression relating $\mathbf{f}_i$ to the combined dynamic properties of links i through N is desired. As illustrated in Figure 25, this relationship is given as follows:

$$\mathbf{f}_i = \mathbf{I}_i^* \mathbf{a}_i - \boldsymbol{\beta}_i^*. \tag{7.6}$$

The matrix, $\mathbf{I}_i^*$, is the ith AB inertia which is the inertia of links i through N that is "felt" at the ith coordinate system when joints $i+1$ to N are free to move (that is, the joint torques/forces are zero). 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 the effects of all the joint torques/forces outboard of this link and a contact force, $\mathbf{f}_{N+1}$, exerted by the end-effector.

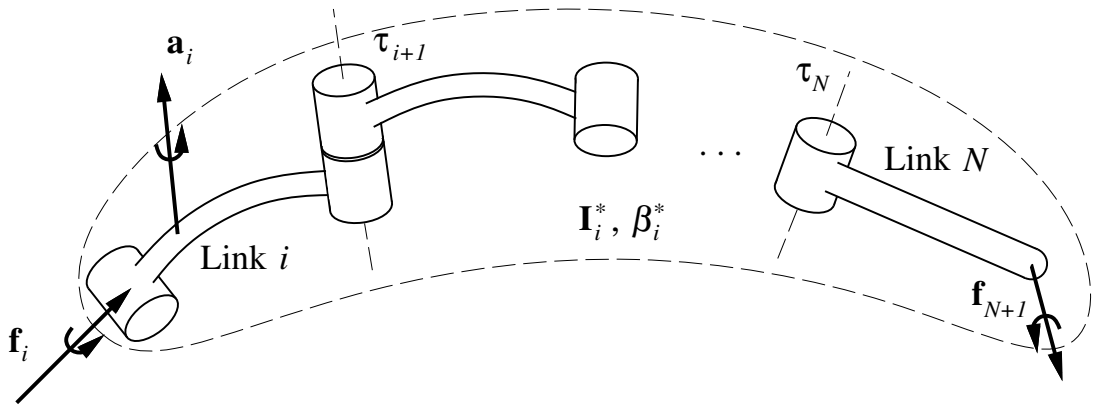

Figure 25: Articulated body dynamics.

Equations needed to compute $\mathbf{I}_i^*$ and $\boldsymbol{\beta}_i^*$ can be found in [15, 17] and a brief derivation using spatial notation is included here. The goal of this derivation is to find expressions for $\mathbf{I}_{i-1}^*$ and $\boldsymbol{\beta}_{i-1}^*$. The first step is to substitute Eq. (7.1) into Eq. (7.6) as follows:

$$\mathbf{f}_i = \mathbf{I}_i^* \left({}^i\mathbf{X}_{i-1}\, \mathbf{a}_{i-1} + \boldsymbol{\phi}_i \ddot{q}_i + \boldsymbol{\zeta}_i \right) - \boldsymbol{\beta}_i^*. \tag{7.7}$$

This force is projected onto the joint axis as in Eq. (7.5) and solved for the unknown joint acceleration, $\ddot{q}_i$. This yields the following equation:

$$\ddot{q}_i = \left(\boldsymbol{\phi}_i^T \mathbf{I}_i^* \boldsymbol{\phi}_i \right)^{-1} \left[\tau_i + \boldsymbol{\phi}_i^T \boldsymbol{\beta}_i^* - \boldsymbol{\phi}_i^T \mathbf{I}_i^* \left({}^i\mathbf{X}_{i-1}\, \mathbf{a}_{i-1} + \boldsymbol{\zeta}_i \right) \right]. \tag{7.8}$$

This is substituted back into Eq. (7.7) to remove the unknown joint acceleration, and the equation that results can be simplified to:

$$\mathbf{f}_i = \mathbf{N}_i\, {}^i\mathbf{X}_{i-1}\, \mathbf{a}_{i-1} - \left[\boldsymbol{\beta}_i^* - \mathbf{N}_i\, \boldsymbol{\zeta}_i - \mathbf{n}_i (m_i^*)^{-1} \tau_i^* \right], \tag{7.9}$$

where

$$\mathbf{n}_i = \mathbf{I}_i^* \boldsymbol{\phi}_i, \tag{7.10}$$

$$m_i^* = \boldsymbol{\phi}_i^T \mathbf{I}_i^* \boldsymbol{\phi}_i, \tag{7.11}$$

$$\mathbf{N}_i = \mathbf{I}_i^* - \mathbf{I}_i^* \boldsymbol{\phi}_i \left(\boldsymbol{\phi}_i^T \mathbf{I}_i^* \boldsymbol{\phi}_i \right)^{-1} \boldsymbol{\phi}_i^T \mathbf{I}_i^*, \quad \text{and} \tag{7.12}$$

$$\tau_i^* = \tau_i + \boldsymbol{\phi}_i^T \boldsymbol{\beta}_i^*. \tag{7.13}$$

The inertia matrix, $\mathbf{N}_i$, is called the *reflected inertia* because it is the AB inertia of links i through N after it has been reflected to the inboard side of joint i when no

joint torque/force is applied and the joint is free to move. The quantity, m_i^*, is the inertial load from the outboard links that is felt by the joint motor. It defines the linear, time-varying relationship between the joint torque/force and the resulting acceleration as shown in Eq. (7.8).

Finally, Eq. (7.9) is substituted into the force balance equation [Eq. (7.3)] for link $i-1$. Grouping terms to put the result into the same form as Eq. (7.6) yields the desired equation:

$$\begin{aligned} \mathbf{f}_{i-1} &= \left(\mathbf{I}_{i-1} + {}^{i}\mathbf{X}_{i-1}^{T}\mathbf{N}_i\,{}^{i}\mathbf{X}_{i-1}\right)\mathbf{a}_{i-1} - \\ &\qquad \left\{\boldsymbol{\beta}_{i-1} + {}^{i}\mathbf{X}_{i-1}^{T}\left[\boldsymbol{\beta}_i^* - \mathbf{N}_i\,\boldsymbol{\zeta}_i - \mathbf{n}_i(m_i^*)^{-1}\tau_i^*\right]\right\}. \end{aligned} \tag{7.14}$$

This completes the derivation, and the recursive equations for the AB inertia and force in Eq. (7.6) are summarized as follows:

$$\mathbf{I}_{i-1}^* = \mathbf{I}_{i-1} + {}^{i}\mathbf{X}_{i-1}^{T}\,\mathbf{N}_i\,{}^{i}\mathbf{X}_{i-1}, \tag{7.15}$$

$$\boldsymbol{\beta}_{i-1}^* = \boldsymbol{\beta}_{i-1} + {}^{i}\mathbf{X}_{i-1}^{T}\left[\boldsymbol{\beta}_i^* - \mathbf{N}_i\,\boldsymbol{\zeta}_i - \mathbf{n}_i(m_i^*)^{-1}\tau_i^*\right]. \tag{7.16}$$

The initial values for this recursion are obtained by directly comparing Eqs. (7.3) and (7.6) for the last link. This implies that:

$$\mathbf{I}_N^* = \mathbf{I}_N, \tag{7.17}$$

$$\boldsymbol{\beta}_N^* = \boldsymbol{\beta}_N - {}^{N+1}\mathbf{X}_N^{T}\,\mathbf{f}_{N+1}, \tag{7.18}$$

where $\mathbf{f}_{N+1}$ is the external force, if present, that is exerted by the last link on the environment.

With expressions for the computation of the AB inertias, $\mathbf{I}_i^*$, and forces, $\boldsymbol{\beta}_i^*$, the AB dynamics algorithm to compute the joint accelerations can be summarized by examining Eq. (7.8) in detail. In this equation, $\boldsymbol{\beta}_i^*$ depends on $\boldsymbol{\beta}_i$ which, along with $\boldsymbol{\zeta}_i$, is a function of link velocities. The computation of the link velocities requires a recursion from the base of the chain to the end-effector and must be completed first. The computation of $\boldsymbol{\beta}_i$ and $\boldsymbol{\zeta}_i$ are also included in this step. Then, a backward recursion is needed to compute $\mathbf{I}_i^*$ and $\boldsymbol{\beta}_i^*$. The final forward recursion to compute the joint accelerations begins with the known base acceleration, $\mathbf{a}_0 = \mathbf{0}$, and evaluates Eqs. (7.8) and (7.1), in that order, from the base of the chain to the end-effector. Using this approach, Appendix B presents an efficient implementation of this procedure for the computation of the dynamics of a serial chain with single degree-of-freedom joints and a fixed base. For a chain with N revolute joints, the algorithm requires [$(224N - 254)$M, $(205N - 241)$A], and when $N = 6$ it requires [1090M, 989A]. This is approximately 15% less than the most efficient algorithm previously reported by Brandl, Johanni, and Otter [17].

7.2.3 Mobile Base and Multiple Chains

When simulating the URV systems discussed at the beginning of Chapter 6, the base of the chain is no longer fixed with respect to an inertial frame and the equations must be augmented to model this characteristic. In the case of the underwater legged vehicle, multiple serial chains attached to this base, resulting in a tree-structured topology, must also be modeled. In this section, the equations needed to extend the AB dynamics algorithm derived in the previous section to model these characteristics are presented. The decoupled tree-structure (DTS) approach to modeling the kinematics of these systems which was presented in Section 5.3.1 for land-based legged vehicles and illustrated in Figure 17 is also used.

Since the base is mobile, a force balance equation for this rigid body is used to relate its unknown acceleration, $\mathbf{a}_r$, with applied forces. First, Eq. (7.3) is augmented to account for multiple links attached to the reference member as follows:

$$\mathbf{f}_r - \sum_{k=1}^{m} {}^r\mathbf{f}_k \;=\; \mathbf{I}_r\,\mathbf{a}_r - \boldsymbol{\beta}_r, \tag{7.19}$$

where $\mathbf{f}_r$ is an external force applied to the reference member and could correspond to thruster forces, and ${}^r\mathbf{f}_k$ is the force exerted by the reference member onto the first link of chain k. It is important to note that ${}^r\mathbf{f}_k$ is the same as $\mathbf{f}_1$ (from the previous section), which has been transformed to the reference member's coordinate system. In this section, $\mathbf{f}_1$ is referred to as $\mathbf{f}_{k,1}$ to distinguish between multiple chains, and ${}^r\mathbf{f}_k$ is a shortened notation for ${}^r\mathbf{f}_{k,1}$.

When using the DTS kinematic modeling, the transformation between the first link of a chain and the reference member is most efficiently implemented with a full set of MDH parameters (planar x and z transformations) from the first link's coordinate system to a base coordinate system attached to the reference member, labeled $(k,0)$, and then a z-planar transformation to the reference member's coordinate system. Therefore, the force balance equation is rewritten as follows:

$$\mathbf{f}_r - \sum_{k=1}^{m} {}^0\mathbf{X}_{k,r}^T\,{}^1\mathbf{X}_{k,0}^T\,\mathbf{f}_{k,1} \;=\; \mathbf{I}_r\,\mathbf{a}_r - \boldsymbol{\beta}_r, \tag{7.20}$$

where ${}^1\mathbf{X}_{k,0}^T$ is the spatial transformation defined by a full set of MDH parameters and ${}^0\mathbf{X}_{k,r}^T$ is the additional z-planar transformation.

The goal of this section is to derive an expression for the reference member's acceleration, $\mathbf{a}_r$, in terms of known quantities. Therefore, the force vectors, $\mathbf{f}_{k,1}$, must be replaced with functions of $\mathbf{a}_r$ and known quantities. The desired expression for this force can be derived from Eq. (7.9) which is rewritten here for the first link of chain k as follows:

$$\mathbf{f}_{k,1} \;=\; \mathbf{N}_{k,1}\,{}^1\mathbf{X}_{k,0}\,\mathbf{a}_{k,0} - \mathbf{f}_{k,1}^*, \tag{7.21}$$

where

$$\mathbf{f}^*_{k,1} \;=\; \boldsymbol{\beta}^*_{k,1} - \mathbf{N}_{k,1}\,\boldsymbol{\zeta}_{k,1} - \mathbf{n}_{k,1}(m^*_{k,1})^{-1}\tau^*_{k,1}. \tag{7.22}$$

This is the component of the force exerted on the link which, through recursion, is a known function of the state of chain k, its joint forces/torques, and the contact force on the tip of the chain. The vector, $\mathbf{a}_{k,0}$, is the spatial acceleration of the base coordinate system of chain k. This is a point fixed to the reference member, so this acceleration can be written in terms of $\mathbf{a}_r$ (shown in Section 5.3.1) as follows:

$$\mathbf{a}_{k,0} \;=\; {}^0\mathbf{X}_{k,r}\,\mathbf{a}_r + \boldsymbol{\zeta}_{k,0}, \tag{7.23}$$

where $\boldsymbol{\zeta}_{k,0}$ is defined similar to Eq. (7.2) as:

$$\boldsymbol{\zeta}_{k,0} \;=\; \begin{bmatrix} \mathbf{0} \\ {}^0\boldsymbol{R}_{k,r}\left[\boldsymbol{\omega}_r \times \left(\boldsymbol{\omega}_r \times {}^r\boldsymbol{p}_{k,0}\right)\right] \end{bmatrix}. \tag{7.24}$$

Therefore, $\mathbf{f}_{k,1}$ is written in terms of $\mathbf{a}_r$ as follows:

$$\mathbf{f}_{k,1} \;=\; \mathbf{N}_{k,1}\,{}^1\mathbf{X}_{k,0}\left({}^0\mathbf{X}_{k,r}\,\mathbf{a}_r + \boldsymbol{\zeta}_{k,0}\right) - \mathbf{f}^*_{k,1}. \tag{7.25}$$

Finally, this is substituted into Eq. (7.20) and the terms multiplying $\mathbf{a}_r$ are grouped together as follows:

$$\begin{aligned} \mathbf{f}_r \;=\; & \left[\mathbf{I}_r + \sum_{k=1}^{m} {}^0\mathbf{X}^T_{k,r}\left({}^1\mathbf{X}^T_{k,0}\mathbf{N}_{k,1}\,{}^1\mathbf{X}_{k,0}\right){}^0\mathbf{X}_{k,r}\right]\mathbf{a}_r - \\ & \boldsymbol{\beta}_r - \sum_{k=1}^{m} {}^0\mathbf{X}^T_{k,r}\left[{}^1\mathbf{X}^T_{k,0}\mathbf{f}^*_{k,1} - \left({}^1\mathbf{X}^T_{k,0}\mathbf{N}_{k,1}\,{}^1\mathbf{X}_{k,0}\right)\boldsymbol{\zeta}_{k,0}\right], \end{aligned} \tag{7.26}$$

which completes the derivation of the equation of motion for the mobile base.

Since $\mathbf{f}_r$, if present, is given in the forward dynamics problem, the AB form of the equation of motion to compute the acceleration of the mobile base in the equation above can be expressed as follows:

$$\mathbf{a}_r \;=\; (\mathbf{I}^*_r)^{-1}\left(\mathbf{f}_r + \boldsymbol{\beta}^*_r\right), \tag{7.27}$$

which was also stated by Featherstone ([27], p. 122). The needed quantities, $\mathbf{I}^*_r$ and $\boldsymbol{\beta}^*_r$, can be determined given the state of the system and joint inputs which are taken from Eq. (7.26) as follows:

$$\mathbf{I}^*_r \;=\; \mathbf{I}_r + \sum_{k=1}^{m} {}^0\mathbf{X}^T_{k,r}\left({}^1\mathbf{X}^T_{k,0}\mathbf{N}_{k,1}\,{}^1\mathbf{X}_{k,0}\right){}^0\mathbf{X}_{k,r}, \tag{7.28}$$

which is the AB inertia of the entire system and expressed with respect to the reference member's coordinate system, and

$$\boldsymbol{\beta}_r^* = \boldsymbol{\beta}_r + \sum_{k=1}^{m} {}^0\mathbf{X}_{k,r}^T \left[{}^1\mathbf{X}_{k,0}^T \mathbf{f}_{k,1}^* - \left({}^1\mathbf{X}_{k,0}^T \mathbf{N}_{k,1} {}^1\mathbf{X}_{k,0} \right) \boldsymbol{\zeta}_{k,0} \right], \tag{7.29}$$

which is the resultant bias force exerted on the reference member from the chains in the system.

7.2.4 Gravity

Up to this point, gravitational effects have been ignored. They can be included in the previous derivations in two ways. Conceptually, the easiest method is to add a gravitational force term to the force balance equation, Eq. (7.3), for each rigid body as follows:

$$\mathbf{f}_i + \mathbf{f}_i^g - {}^{i+1}\mathbf{X}_i^T \mathbf{f}_{i+1} = \mathbf{I}_i \mathbf{a}_i - \boldsymbol{\beta}_i. \tag{7.30}$$

The force due to gravity can be computed for the body as follows:

$$\mathbf{f}_i^g = \mathbf{I}_i {}^i\mathbf{a}_g, \tag{7.31}$$

where ${}^i\mathbf{a}_g$ is the gravitational acceleration expressed in the body's coordinate system and consists of only a translational component:

$${}^i\mathbf{a}_g = \begin{bmatrix} \mathbf{0} \\ {}^i\boldsymbol{a}_g \end{bmatrix}. \tag{7.32}$$

The simplest way to incorporate this into the previous derivations is to redefine $\boldsymbol{\beta}_i$ to include this force, and the appearance of the previous derivations will remain unchanged. From a computational efficiency standpoint, the disadvantage to this approach is that the gravitational acceleration must be determined with respect to each rigid body's coordinate system.

The alternate approach, which is also used in efficient inverse dynamics algorithms, is to bias the spatial acceleration of each link by negative gravitational acceleration. This is possible by substituting Eq. (7.31) into Eq (7.30) and collecting the terms multiplying the spatial inertia:

$$\mathbf{f}_i - {}^{i+1}\mathbf{X}_i^T \mathbf{f}_{i+1} = \mathbf{I}_i \left(\mathbf{a}_i - {}^i\mathbf{a}_g \right) - \boldsymbol{\beta}_i \tag{7.33}$$

$$= \mathbf{I}_i \mathbf{a}_i' - \boldsymbol{\beta}_i. \tag{7.34}$$

This becomes the replacement for Eq. (7.3) in the previous derivations. The propagation of this acceleration is accomplished with Eq. (7.1):

$$\mathbf{a}_i' = {}^i\mathbf{X}_{i-1}\, \mathbf{a}_{i-1}' + \boldsymbol{\phi}_i \ddot{q}_i + \boldsymbol{\zeta}_i. \tag{7.35}$$

Therefore, the previous derivations proceed as before except every occurrence of spatial acceleration, $\mathbf{a}$, is changed to the biased spatial acceleration, $\mathbf{a}'$.

The only change made to the single chain algorithm with a fixed base is to bias the zero base acceleration as follows:

$$\mathbf{a}'_0 = -\begin{bmatrix} \mathbf{0} \\ {}^0\boldsymbol{a}_g \end{bmatrix}. \tag{7.36}$$

Since the base coordinate system is fixed, this quantity is constant and can be computed off line. The rest of the algorithm remains unchanged and is presented in Appendix B.

For the multiple chain system, the reference member acceleration computation in Eq. (7.27) will now yield a biased result:

$$\mathbf{a}'_r = (\mathbf{I}^*_r)^{-1}(\mathbf{f}_r + \boldsymbol{\beta}^*_r). \tag{7.37}$$

This result is used in the Forward Acceleration recursion for each chain to compute joint accelerations. For the purpose of simulation, the unbiased reference member acceleration must also be computed as follows:

$$\mathbf{a}_r = \mathbf{a}'_r + \begin{bmatrix} \mathbf{0} \\ {}^r\boldsymbol{R}_e\,{}^e\boldsymbol{a}_g \end{bmatrix}, \tag{7.38}$$

where ${}^e\boldsymbol{a}_g$ is the constant Cartesian vector of gravitational acceleration expressed in the *e*arth (inertial) coordinate system, and ${}^r\boldsymbol{R}_e$ is the rotation matrix describing the orientation of the reference member with respect to the inertial coordinate system.

7.2.5 Algorithm Summary

Finally, the development of the dynamics algorithm for a system with multiple chains and a mobile base on land can be completed. It uses the single chain algorithm in Appendix B which is performed for each serial chain that is attached to the reference member. However, the derivation for the mobile base implies a number of changes that must be made to the three recursions in this algorithm.

Before the Forward Kinematics recursion is executed for each chain, the angular velocity of the reference member, $\boldsymbol{\omega}_r$, must be transformed to the base coordinate system of each chain to obtain $\boldsymbol{\omega}_{k,0}$. This quantity becomes the starting condition of this recursion, and replaces $\boldsymbol{\omega}_0 = \mathbf{0}$ in the fixed base algorithm. The vector, $\boldsymbol{\zeta}_{k,0}$, from Eq. (7.24) is added to the beginning of this recursion for each chain, and the computation of $\boldsymbol{\beta}_r$ is performed before these recursions.

The end of the Backward Dynamics recursion for each chain must include the necessary computations to obtain $\mathbf{I}^*_r$ and $\boldsymbol{\beta}^*_r$. The former involves performing congruence transformations on the reflected inertia, $\mathbf{N}_1$, to express it in the reference

member's coordinate system and shown in the rightmost term on Eq. (7.28). Computation of $\boldsymbol{\beta}_r^*$ requires the computation of the bias forces exerted on the reference member from each chain as expressed in the Eq. (7.29).

Before the Forward Accelerations recursion for each chain can proceed, the biased reference member acceleration, $\mathbf{a}_r'$, must be computed as shown by Eq. (7.37). Then, $\mathbf{a}_{k,0}'$ from Eq. (7.23) must be determined which becomes the initial value of this recursion for each chain. In addition, the unbiased reference member acceleration, $\mathbf{a}_r$, from Eq. (7.38) must be determined.

The resulting algorithm is called the Articulated Body/Decoupled Tree-Structure (AB/DTS) algorithm which was first developed by Freeman and Orin [7]. A detailed listing of this algorithm is not included here but can be easily extracted from the version with hydrodynamics which is developed in the next section and listed in Tables 13–15. The computational requirements for the more efficient algorithm derived here is shown in Table 16. For an m-chain system with N degrees of freedom per chain, the algorithm requires [$(224mN+59m+99)$M, $(205mN+63m+68)$A]. For the land-based, legged vehicle discussed in Chapter 5, with $m = 6$ and $N = 3$, this corresponds to [4485M, 4136A] which is slightly more than the CRB/DTS algorithm. The AB/DTS algorithm is more efficient when $N > 3$.

7.3 Incorporating Hydrodynamic Effects

The final and most important step in the derivation of the underwater dynamics algorithm for URV systems is to incorporate the computation of hydrodynamic forces on a single rigid body that was developed in the previous chapter into the algorithm developed above. First, the spatial force balance equation for a single link that is submerged in a fluid is developed which includes these hydrodynamic forces. The differences between this and the equation used in the previous derivation provide the information necessary to modify the AB algorithm to simulate URV systems.

Including the link-to-link forces, gravity and hydrodynamic effects, the force balance on link i can be written as follows:

$$\mathbf{f}_i - {}^{i+1}\mathbf{X}_i^T\, \mathbf{f}_{i+1} + \mathbf{f}_i^A + \mathbf{f}_i^D + \mathbf{f}_i^{TB} \;=\; \mathbf{I}_i\, \mathbf{a}_i' - \boldsymbol{\beta}_i, \tag{7.39}$$

where, again:

$$\mathbf{a}_i' \;=\; \mathbf{a}_i - {}^i\mathbf{a}_g. \tag{7.40}$$

The added mass force, $\mathbf{f}_i^A$, is also a function of the unknown link acceleration, $\mathbf{a}_i$. Substituting in from Eq. (6.20) and rearranging the terms leads to the following equation:

$$\mathbf{f}_i - {}^{i+1}\mathbf{X}_i^T\, \mathbf{f}_{i+1} \;=\; \mathbf{I}_i\mathbf{a}_i' + \mathbf{I}_i^A\mathbf{a}_i - \boldsymbol{\beta}_i - \boldsymbol{\beta}_i^A - \mathbf{f}_i^D - \mathbf{f}_i^{TB}. \tag{7.41}$$

Since the efficient AB algorithm is implemented using the biased spatial acceleration, $\mathbf{a}'_i$, this equation must be modified to remove $\mathbf{a}_i$. This is accomplished by substituting for $\mathbf{a}_i$ from Eq. (7.40):

$$\mathbf{f}_i - {}^{i+1}\mathbf{X}_i^T\, \mathbf{f}_{i+1} \;=\; \mathbf{I}_i\mathbf{a}'_i + \mathbf{I}_i^A\left(\mathbf{a}'_i + {}^i\mathbf{a}_g\right) - \boldsymbol{\beta}_i - \boldsymbol{\beta}_i^A - \mathbf{f}_i^D - \mathbf{f}_i^{TB}. \tag{7.42}$$

The terms multiplying the biased acceleration are combined and the new term, $\mathbf{I}_i^A\ {}^i\mathbf{a}_g$, is most efficiently included with the added mass bias force, $\boldsymbol{\beta}_i^A$, from Eq. (6.21). The resulting equation is given as:

$$\mathbf{f}_i - {}^{i+1}\mathbf{X}_i^T\, \mathbf{f}_{i+1} \;=\; \left(\mathbf{I}_i + \mathbf{I}_i^A\right)\mathbf{a}'_i - \boldsymbol{\beta}_i - \boldsymbol{\beta}_i^{A'} - \mathbf{f}_i^D - \mathbf{f}_i^{TB}. \tag{7.43}$$

where

$$\boldsymbol{\beta}_i^{A'} \;=\; \boldsymbol{\beta}_i^A - \mathbf{I}_i^A \begin{bmatrix} \mathbf{0} \\ {}^i\boldsymbol{a}_g \end{bmatrix} \tag{7.44}$$

$$\;=\; \mathbf{I}_i^A \begin{bmatrix} \mathbf{0} \\ ({}^i\boldsymbol{a}_f - {}^i\boldsymbol{a}_g) + \boldsymbol{\omega}_i \times \boldsymbol{v}_i^r \end{bmatrix} - \begin{bmatrix} \tilde{\boldsymbol{\omega}}_i & \tilde{\boldsymbol{v}}_i^r \\ \mathbf{0} & \tilde{\boldsymbol{\omega}}_i \end{bmatrix} \mathbf{I}_i^A \begin{bmatrix} \boldsymbol{\omega}_i \\ \boldsymbol{v}_i^r \end{bmatrix}. \tag{7.45}$$

Combining the terms in this fashion results in no additional computation because the quantity, ${}^i\boldsymbol{a}_f - {}^i\boldsymbol{a}_g$, is already obtained in the computation of the total buoyancy force, $\mathbf{f}_i^{TB}$.

Rewriting the equation to resemble Eq. (7.33) leads to the following hydrodynamic equation of motion for the link:

$$\mathbf{f}_i - {}^{i+1}\mathbf{X}_i^T\mathbf{f}_{i+1} \;=\; \mathbf{I}_i^H\mathbf{a}'_i - \boldsymbol{\beta}_i^H, \tag{7.46}$$

where

$$\mathbf{I}_i^H \;=\; \mathbf{I}_i + \mathbf{I}_i^A, \quad \text{and} \tag{7.47}$$

$$\boldsymbol{\beta}_i^H \;=\; \boldsymbol{\beta}_i + \boldsymbol{\beta}_i^{A'} + \mathbf{f}_i^D + \mathbf{f}_i^{TB}. \tag{7.48}$$

The consequence of this equation is that the spatial acceleration of the body and the resultant force are now linearly related through the sum of the rigid body's spatial inertia and the added mass of the fluid surrounding the body. The hydrodynamic AB algorithm can be derived as before by using Eq. (7.46) in place of Eq. (7.33) [or Eq. (7.3)]. Since the form of this equation is the same, the derivation will proceed as before, and the resulting efficient algorithm is listed in Tables 13–15 for multiple chain URVs.

A comparison of Eqs. (7.46) and (7.33) indicates two changes that are to be made to the resulting algorithm to incorporate the hydrodynamic effects. First, the use of each link's spatial inertia, $\mathbf{I}_{k,i}$, is replaced with its "hydrodynamic inertia," $\mathbf{I}_{k,i}^H$. The second change replaces $\boldsymbol{\beta}_{k,i}$ with $\boldsymbol{\beta}_{k,i}^H$. Both of these steps have been underlined in

Table 13: Articulated Body algorithm for multiple-chain URV systems.

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

$$
\begin{aligned}
\boldsymbol{v}_r^r &= \boldsymbol{v}_r - {}^r\boldsymbol{R}_e\,{}^e\boldsymbol{v}_f \\
{}^r\boldsymbol{a}_g &= {}^r\boldsymbol{R}_e(g\hat{\mathbf{z}}_e) \qquad \text{where } \hat{\mathbf{z}}_e = [0\ 0\ 1]^T \\
{}^r\boldsymbol{a}_{f-g} &= {}^r\boldsymbol{R}_e\,{}^e\boldsymbol{a}_f - {}^r\boldsymbol{a}_g \\
\boldsymbol{\beta}_r^H &= -\begin{bmatrix} \boldsymbol{\omega}_r \times \bar{\boldsymbol{I}}_r\boldsymbol{\omega}_r \\ \mathbf{0} \end{bmatrix} + \mathbf{f}_r^D(\boldsymbol{\omega}_r, \boldsymbol{v}_r^r) + m_r^f \begin{bmatrix} \boldsymbol{b}_r \times {}^r\boldsymbol{a}_{f-g} \\ {}^r\boldsymbol{a}_{f-g} \end{bmatrix} + \\
&\qquad \mathbf{I}_r^A \begin{bmatrix} \mathbf{0} \\ {}^r\boldsymbol{a}_{f-g} + \boldsymbol{\omega}_r \times \boldsymbol{v}_r^r \end{bmatrix} - \begin{bmatrix} \tilde{\boldsymbol{\omega}}_r & \tilde{\boldsymbol{v}}_r^r \\ \mathbf{0} & \tilde{\boldsymbol{\omega}}_r \end{bmatrix} \mathbf{I}_r^A \begin{bmatrix} \boldsymbol{\omega}_r \\ \boldsymbol{v}_r^r \end{bmatrix}
\end{aligned}
$$

for all $k = 1, \ldots, m$. Given: ${}^0\boldsymbol{R}_{k,r}$, ${}^r\boldsymbol{p}_{k,0}$.

$$
\begin{aligned}
\begin{bmatrix} \boldsymbol{\omega}_{k,0} \\ \boldsymbol{v}_{k,0}^r \end{bmatrix} &= {}^0\mathbf{X}_{k,r} \begin{bmatrix} \boldsymbol{\omega}_r \\ \boldsymbol{v}_r^r \end{bmatrix} \\
{}^0\boldsymbol{a}_{k,f-g} &= {}^0\boldsymbol{R}_{k,r}\,{}^r\boldsymbol{a}_{f-g} \\
\boldsymbol{\zeta}_{k,0} &= \begin{bmatrix} \mathbf{0} \\ {}^0\boldsymbol{R}_{k,r}\left[\boldsymbol{\omega}_r \times \left(\boldsymbol{\omega}_r \times {}^r\boldsymbol{p}_{k,0}\right)\right] \end{bmatrix}
\end{aligned}
$$

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

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

end for i.

end for all k.

Table 14: Articulated Body algorithm for multiple-chain URV systems (continued).

II. Backward Dynamics

for all $k = 1, \ldots, m$. Given: ${}^{N}\mathbf{N}_{k,N+1} = \mathbf{0}$, ${}^{N}\mathbf{f}^{*}_{k,N+1} = -{}^{N+1}\mathbf{X}^{T}_{N}\ \mathbf{f}_{N+1}$ (tip forces).

for $i = N, .., 1$. Given: $\tau_{k,i}$.

$$
\begin{aligned}
\mathbf{I}^{*}_{k,i} &= \underline{\mathbf{I}^{H}_{k,i}} + {}^{i}\mathbf{N}_{k,i+1} \\
\mathbf{n}_{k,i} &= \mathbf{I}^{*}_{k,i}\boldsymbol{\phi}_{k,i} \\
m^{*}_{k,i} &= \boldsymbol{\phi}^{T}_{k,i}\mathbf{I}^{*}_{k,i}\boldsymbol{\phi}_{k,i} \\
\mathbf{N}_{k,i} &= \mathbf{I}^{*}_{k,i} - \mathbf{n}_{k,i}(m^{*}_{k,i})^{-1}\mathbf{n}^{T}_{k,i} \\
\boldsymbol{\beta}^{*}_{k,i} &= \underline{\boldsymbol{\beta}^{H}_{k,i}} + {}^{i}\mathbf{f}^{*}_{k,i+1} \\
\tau^{*}_{k,i} &= \boldsymbol{\phi}^{T}_{k,i}\boldsymbol{\beta}^{*}_{k,i} + \tau_{k,i} \\
{}^{i-1}\mathbf{N}_{k,i} &= {}^{i}\mathbf{X}^{T}_{k,i-1}\mathbf{N}_{k,i}{}^{i}\mathbf{X}_{k,i-1} \\
{}^{i-1}\mathbf{f}^{*}_{k,i} &= {}^{i}\mathbf{X}^{T}_{k,i-1}\left[\boldsymbol{\beta}^{*}_{k,i} - \mathbf{N}_{k,i}\boldsymbol{\zeta}_{k,i} - \mathbf{n}_{k,i}(m^{*}_{k,i})^{-1}\tau^{*}_{k,i}\right]
\end{aligned}
$$

end for i.

$$
\begin{aligned}
{}^{r}\mathbf{N}_{k,1} &= {}^{0}\mathbf{X}^{T}_{k,r}\,{}^{0}\mathbf{N}_{k,1}{}^{0}\mathbf{X}_{k,r} \\
{}^{r}\mathbf{f}^{*}_{k,1} &= {}^{0}\mathbf{X}^{T}_{k,r}\left({}^{0}\mathbf{f}^{*}_{k,1} - {}^{0}\mathbf{N}_{k,1}\boldsymbol{\zeta}_{k,0}\right)
\end{aligned}
$$

end for all k.

---

the Backward Dynamics recursion of Table 14, and in the computation of $\mathbf{a}'_r$ in the Forward Acceleration recursion of Table 15. The use of the alternative terms has little effect on the computation requirements of these recursions of the AB algorithm as will be shown in the next section.

It is the computation of $\boldsymbol{\beta}^{H}_{k,i}$ itself that adds a significant amount to the Forward Kinematics recursion in Table 13. In this recursion, the computation of $\boldsymbol{\beta}_{k,i}$ is replaced with the computation of $\boldsymbol{\beta}^{H}_{k,i}$. While this equation still computes $\boldsymbol{\beta}_{k,i}$ (the first term of the equation in the table), the hydrodynamic forces and added mass bias term are also included. Computation of the total buoyancy force, $\mathbf{f}^{TB}_{k,i}$, requires computation of ${}^{i}\boldsymbol{a}_f - {}^{i}\boldsymbol{a}_g$.

This term is most efficiently computed by performing the subtraction once at the

Table 15: Articulated Body algorithm for multiple-chain URV systems (continued).

III. Forward Accelerations Given: $\mathbf{f}_r$.

$$\mathbf{a}'_r = \left[\underline{\mathbf{I}_r^H} + \sum_{k=1}^{m} {}^r\mathbf{N}_{k,1}\right]^{-1} \left[\mathbf{f}_r + \underline{\boldsymbol{\beta}_r^H} + \sum_{k=1}^{m} {}^r\mathbf{f}^*_{k,1}\right]$$

$$\mathbf{a}_r = \mathbf{a}'_r + \begin{bmatrix} \mathbf{0} \\ {}^r\boldsymbol{a}_g \end{bmatrix}$$

for all $k = 1, \ldots, m$. Given: ${}^0\mathbf{X}_{k,r}$.

$$\mathbf{a}'_{k,0} = {}^0\mathbf{X}_{k,r}\mathbf{a}'_r + \boldsymbol{\zeta}_{k,0}$$

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

$$\bar{\mathbf{a}}_{k,i} = {}^i\mathbf{X}_{k,i-1}\mathbf{a}'_{k,i-1} + \boldsymbol{\zeta}_{k,i}$$
$$\ddot{q}_{k,i} = (m^*_{k,i})^{-1}\tau^*_{k,i} - \left[\mathbf{n}_{k,i}(m^*_{k,i})^{-1}\right]^T \bar{\mathbf{a}}_{k,i}$$
$$\mathbf{a}'_{k,i} = \bar{\mathbf{a}}_{k,i} + \boldsymbol{\phi}_{k,i}\ddot{q}_{k,i}$$

end for i.

end for all k.

---

reference member coordinate system and then transforming a single vector (referred to as ${}^i\boldsymbol{a}_{k,f-g}$ in the table) to successive coordinate systems. The computation of the drag force, $\mathbf{f}^D_{k,i}$, requires the computation of the linear velocity of the body relative to the fluid, $\boldsymbol{v}^r_{k,i}$. This is combined with the angular velocity of the bodies, and the resulting spatial velocity vector is transformed in one operation, rather than transforming the two separately. Computation of the added mass bias force, $\boldsymbol{\beta}_i^{A'}$, requires both of these terms. To obtain them with respect to each link's coordinate system, their computation is most efficiently incorporated into the Forward Kinematics recursion as shown in Table 13.

7.4 Computational Requirements

In this section the computational requirements for this implementation of the AB/-DTS algorithm are determined. Both land-based and underwater versions are examined, and the results are listed in Table 16. In the top section of the table, the requirements for the Forward Kinematics recursion are listed. In this step, velocity and acceleration terms are determined with respect to the reference member's coordinate system and then this computation propagates along each chain. Terms dependent on these quantities are also computed for each body in the system.

In this implementation, the orientation of the reference member is specified by three Euler angles [60], which is represented by the rotation matrix, ${}^{r}\boldsymbol{R}_{e}$. To achieve the most efficient transformation of these quantities, this is stored as three separate planar rotations which requires no multiplications or additions to compute (only sines and cosines of each angle). With this form, the rotations of Cartesian vectors from inertial to body-fixed frames is accomplished most efficiently with three successive planar rotations each requiring [4M, 2A] (see Appendix A). For the underwater algorithm, the vector rotations of fluid velocity and acceleration cost [12M, 6A] each, and therefore, the cost of computing $\boldsymbol{v}_{r}^{r}$ or ${}^{r}\boldsymbol{a}_{f-g}$ is [12M, 9A]. Both versions of the algorithm require the computation of the gravitational acceleration, ${}^{r}\boldsymbol{a}_{g}$, which only requires [4M, 0A] when using Z-Y-X Euler angles with the inertial z-axis pointing "down."

Finally, $\boldsymbol{\beta}_{r}$ or $\boldsymbol{\beta}_{r}^{H}$, for the land-based or underwater algorithm, respectively, can be computed. Using the DTS kinematic modeling, the reference member coordinate system is coincident with its principal axes which results in a diagonal spatial inertia matrix and a simpler computation ([9M, 3A]) for $\boldsymbol{\beta}_{r}$ because $\boldsymbol{h}_{r}$ is zero and $\bar{\boldsymbol{I}}_{r}$ is diagonal. This is also the first term of $\boldsymbol{\beta}_{r}^{H}$, and to complete its computation requires an additional [133M, 107A] for the hydrodynamic terms.

Computational requirements for $\boldsymbol{\omega}_{k,0}$, $\boldsymbol{v}_{k,0}^{r}$, ${}^{0}\boldsymbol{a}_{k,f-g}$, and $\boldsymbol{\zeta}_{k,0}$ are straightforward, realizing that the transformation between the reference member coordinate system and the base coordinate system for each chain involves a single z-planar screw which also implies that ${}^{r}\boldsymbol{p}_{k,0}$ has one nonzero element. Compared to the fixed base algorithm in the appendix, the angular velocity, $\boldsymbol{\omega}_{k,0}$, is now the starting value for the recursion along the chains. It is no longer zero in the case of a mobile base; therefore, all of the reductions in computation in the first few iterations of the fixed base algorithm, because of these zero elements, cannot be utilized here. Finally, $\boldsymbol{\beta}_{k,i}$ requires [27M, 15A], and the computation of $\boldsymbol{\beta}_{k,i}^{H}$ requires an additional [133M, 110A] which accounts for about 90% of the additional computation for the underwater algorithm. The computation of additional velocity and acceleration terms account for nearly all of the rest.

The second section of the table lists the computational requirements for the

Table 16: Computational requirements of the multiple-chain URV simulation algorithm.

		Land Based		Underwater	
		$\times$	$+$	$\times$	$+$
I.	$\boldsymbol{v}_r^r$	—	—	12	9
	${}^r\boldsymbol{a}_g$	4	—	4	—
	${}^r\boldsymbol{a}_{f-g}$	—	—	12	9
	$\boldsymbol{\beta}_r/\boldsymbol{\beta}_r^H$	9	3	142	110
	$\boldsymbol{\omega}_{k,0}/\mathbf{v}_{k,0}^r$	$4m$	$2m$	$10m$	$6m$
	${}^0\boldsymbol{a}_{k,f-g}$	—	—	$4m$	$2m$
	$\boldsymbol{\zeta}_{k,0}$	$10m$	$3m$	$10m$	$3m$
	$\boldsymbol{\omega}_{k,i}/\mathbf{v}_{k,i}^r$	$8mN$	$5mN$	$20mN$	$13mN$
	${}^i\boldsymbol{a}_{k,f-g}$	—	—	$8mN$	$4mN$
	$\boldsymbol{\zeta}_{k,i}$	$22mN$	$10mN$	$22mN$	$10mN$
	$\boldsymbol{\beta}_{k,i}/\boldsymbol{\beta}_{k,i}^H$	$27mN$	$15mN$	$160mN$	$125mN$
II.	$\mathbf{I}_{k,i}^*$	—	$15m(N-1)$	—	$21m(N-1)$
	$\mathbf{n}_{k,i}$	—	—	—	—
	$(m_{k,i}^*)^{-1}$	$m(N-1)$	—	$m(N-1)$	—
	$\mathbf{n}_{k,i}\,(m_{k,i}^*)^{-1}$	$5m(N-1)$	—	$5m(N-1)$	—
	$\mathbf{N}_{k,i}$	$15m(N-1)$	$15m(N-1)$	$15m(N-1)$	$15m(N-1)$
	$\boldsymbol{\beta}_{k,i}^*$	—	$6mN$	—	$6mN$
	$\tau_{k,i}^*$	—	mN	—	mN
	${}^{i-1}\mathbf{N}_{k,i}$	$m(70N-10)$	$m(71N-10)$	$70mN$	$71mN$
	${}^{i-1}\mathbf{f}_{k,i}^*$	$m(50N-4)$	$m(43N-4)$	$50mN$	$43mN$
	${}^r\mathbf{N}_{k,1}$	$42m$	$43m$	$42m$	$43m$
	${}^r\mathbf{f}_{k,1}^*$	$28m$	$24m$	$28m$	$24m$
III.	$\mathbf{a}_r'$	86	$27m+62$	86	$27m+71$
	$\mathbf{a}_r$	—	3	—	3
	$\mathbf{a}_{k,0}'$	$10m$	$9m$	$10m$	$9m$
	$\bar{\mathbf{a}}_{k,i}$	$20mN$	$17mN$	$20mN$	$17mN$
	$\ddot{q}_{k,i}$	$6mN$	$6mN$	$6mN$	$6mN$
	$\mathbf{a}_{k,i}'$	—	$m(N-1)$	—	$m(N-1)$
Total:		$224mN+$ $59m+99$	$205mN+$ $63m+68$	$377mN+$ $83m+256$	$333mN+$ $77m+202$
$m,N=6,3$		4485	4136	7540	6658
$m,N=1,6$		1502	1359	2601	2277

Backward Dynamics recursion. Most of the numbers for this recursion are easily derived from the single chain algorithm in Appendix B; however, the computation in this step has been substantially rearranged from the fixed base algorithm in the appendix. Many of the computational requirements are still the same. Since ${}^{N}\mathbf{N}_{k,N+1} = \mathbf{0}$, this implies that $\mathbf{I}^*_{k,N}$, $\mathbf{n}_{k,N}$, $m^*_{k,N}$, $\mathbf{N}_{k,N}$, $(m^*_{k,N})^{-1}$ and $\mathbf{n}_{k,N}\ (m^*_{k,N})^{-1}$ are all constant and can be computed off-line. Unlike the fixed base algorithm, $\mathbf{N}_{k,1}$, ${}^{0}\mathbf{N}_{k,1}$, and ${}^{0}\mathbf{f}^*_{k,1}$ have been added to this recursion. To complete the computation the latter two must also be transformed to the reference member coordinate system. The constant z-planar congruence transform of ${}^{0}\mathbf{N}_{k,1}$ was shown in Appendix A to require [42M, 43A], and the computation of ${}^{r}\mathbf{f}^*_{k,1}$ requires [28M, 24A] per chain.

Computationally, the only difference in this recursion between land-based and the underwater versions of the algorithms involves the computation of the AB inertias, $\mathbf{I}^*_{k,i}$. The addition to obtain $\mathbf{I}^H_{k,i}$ does not add to the amount of computation because both components are constant and the sum is computed off-line. However, the result is a full symmetric matrix which contains six more nonzero elements in the upper triangular portion than the spatial rigid body inertia, $\mathbf{I}_{k,i}$ which increases the requirements to compute $\mathbf{I}^*_{k,i}$ by as many additions. Since $\mathbf{I}^*_{k,N}$ is also a full symmetric matrix, $\mathbf{N}_{k,N}$ no longer has two extra zero elements, and the additional efficiencies found in the computation of ${}^{N-1}\mathbf{N}_{k,N}$ and ${}^{N-1}\mathbf{f}^*_{k,N}$ cannot be implemented.

In the third section of the table, the computational requirements for the Forward Accelerations recursion are listed. To construct the 6×6 system of equations to compute the biased reference member acceleration requires [0M, $(27m - 3)$A] for the land-based algorithm and an additional [0M, 9A] for the underwater algorithm due to additional nonzero elements in $\mathbf{I}^H_r$ and $\boldsymbol{\beta}^H_r$. The resulting matrix is positive definite, so its solution can be completed using Cholesky decomposition and back substitution which requires [86M, 65A] as was determined in Part I for a 6×6 system of equations. The true acceleration of the reference member, is obtained by adding gravitational acceleration to this result ([0M, 3A]).

Unlike the fixed-base algorithm in Appendix B, the biased reference member acceleration is a general six-vector, and additional reductions in computation during the first few iterations of the forward recursions along the chains cannot be employed. Otherwise the amount of computation in the recursion is similar to the fixed base algorithm, and is the same for both land-based and underwater versions. Note that $\mathbf{a}'_{k,N}$ is not needed.

The resulting AB algorithm for land-based systems requires [$(224mN + 59m + 99)$M, $(205mN + 63m + 68)$A] for an m-chain system with N revolute joints each. For the OSU Hexapod or the Aquarobot on land ($m = 6,\ N = 3$), this corresponds to [4485M, 4136A] which is slightly more than the CRB/DTS method that was developed in Chapter 5. For a mobile base system with a 6 DOF arm ($m = 1$, $N = 6$), this corresponds to [1502M, 1359A] which is much less than the CRB/DTS

approach.

When hydrodynamics is introduced, the hydrodynamic inertia matrices destroy the special structure of the rigid body spatial inertias. Since the CRB/DTS algorithms depend heavily on this structure to achieve efficient congruence transformations (see Appendix A), a hydrodynamic version of this algorithm is much less efficient than the AB/DTS algorithm developed here. It has little effect on the AB/DTS algorithm because the AB inertia matrices, to which they are added, are already full symmetric matrices. The computational requirements for the underwater AB/DTS algorithm are $[(377mN + 83m + 256)\text{M}, (333mN + 77m + 202)\text{A}]$. For the Aquarobot this corresponds to [7540M, 6658A], and for *Tiburon* it is [2601M, 2277A]. Compared to the land-based algorithm, hydrodynamics increases the computation between 65% and 70% depending on the topology.

7.5 Efficient ROV/Manipulator System Simulation

Thus far, an AB simulation algorithm has been developed for mobile base systems with multiple chains. The DTS kinematic modeling scheme was employed to cope with general connections of multiple chains to the base in an efficient manner. Another prevalent topology in URV systems contains a single serial chain of rigid bodies, which arises with an underwater vehicle with a single manipulator. One example is *Tiburon* with the Schilling manipulator attached. In this case, the simulation algorithm can be modified specifically for this topology in order to reduce the amount of computation. In this section, more efficient algorithms for such systems are described. In addition, a simplified dynamics algorithm, called decoupled dynamics, to further reduce the amount of computation is examined for these systems.

7.5.1 Full Dynamics: Efficient Kinematic Modeling

With a single chain, the coordinate system attached to the reference member can be placed in any convenient location. Since the goal is to minimize computation, the best location minimizes transformations between the first link of the chain and the reference member. The obvious choice is to place the origin and z-axis of the reference member's coordinate system coincident with the first link's coordinate system. To simplify the notation, the reference member is called link 0 in the serial chain and reference member quantities will be indicated with a trailing subscript of 0. The resulting algorithm is listed in Tables 17 and 18. The computations have been rearranged from the multiple chain algorithm of the previous sections. Besides the additional hydrodynamics computations, it is very similar to the single chain, fixed-base algorithm that was presented in Appendix B. The computational

requirements for this algorithm are listed in Table 19 at the end of this section under the column labeled "Full Dynamics."

In the Forward Kinematics step, the orientation of link 0, the reference member, is still specified with three Euler angles so the computation of $\boldsymbol{v}_0^r$, ${}^0\boldsymbol{a}_g$, and ${}^0\boldsymbol{a}_{f-g}$ remains the same. Because the reference member's coordinate system is no longer at its center of gravity, the computation requirements for $\boldsymbol{\beta}_0^H$ is the same as the other links, [160M, 125A], which is more than in the previous section. However, the computational efficiency gained with this approach becomes apparent with the forward recursion along the chain. Multiple transformations no longer exist between the coordinate systems of the reference member and first link; therefore, computations for intermediate quantities between these, such as $\mathbf{v}_{k,0}^r$, ${}^0\boldsymbol{a}_{k,f-g}$, and $\boldsymbol{\zeta}_{k,0}$, are no longer needed.

Additional efficiencies are gained in the transformation of quantities between coordinate systems 0 and 1. Because of coordinate system 0's placement, the transformation only consists of a rotation about the z-axis when the first joint is revolute, or a translation along the z-axis when it is prismatic. The former case is considered in this discussion so that comparisons can be made with previous results using the same type of joint. Results in Appendix A shows that with a z-planar rotation, the transformation of a Cartesian vector requires [4M, 2A], and the transformation of a spatial vector becomes equivalent to planar rotations of two Cartesian vectors ([8M, 4A]). Because ${}^0\boldsymbol{p}_1$ is also zero, the computation of $\boldsymbol{\zeta}_1$ requires only [2M, 0A] and has only two nonzero elements, and the congruence transformation of the reflected AB inertia, $\mathbf{N}_1$, can be shown to require only [22M, 25A] and the result still contains a zero row and column. These conditions affect the computational requirements for $\mathbf{v}_1^r$, ${}^1\boldsymbol{a}_{f-g}$, $\boldsymbol{\zeta}_1$, $\mathbf{I}_0^*$, $\boldsymbol{\beta}_0^*$, and $\bar{\mathbf{a}}_1$ as shown in the table.

The resulting algorithm requires $[(377N + 130)\text{M}, (333N + 93)\text{A}]$. For *Tiburon* with a Schilling manipulator, $N = 6$ and it requires [2392M, 2091A]. The general algorithm requires [2601M, 2277A] for the same system. Therefore, the algorithm developed in this section specifically for a single chain system results in 8% less computation.

7.5.2 Decoupled Dynamics: Simplified Dynamics Modeling

A second method of reducing the requirements of the dynamics computation involves simplification of the dynamic equations themselves. The approach used thus far in the chapter assumes that the acceleration of the reference member is significantly affected by the forces exerted by the serial chain attached to it. In this case, this acceleration is unknown and coupled to the dynamics computation for the manipulator. This is especially true for the legged vehicles in which nearly all of the

Table 17: Articulated Body algorithm for a URV with a single manipulator.

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

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

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

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

end for i.

Table 18: Articulated Body algorithm for a URV with a single manipulator (continued).

II. Backward Dynamics Given: $\mathbf{I}_N^* = \mathbf{I}_N^H$, $\boldsymbol{\beta}_N^* = \boldsymbol{\beta}_N^H - {}^{N+1}\mathbf{X}_N^T\, \mathbf{f}_{N+1}$.

for $i = N, \ldots, 1$. Given: τ_i.

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

end for i.

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

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

$$
\mathbf{a}_0 = \mathbf{a}_0' + \begin{bmatrix} \mathbf{0} \\ {}^0\boldsymbol{a}_g \end{bmatrix}
$$

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.

driving force on the reference member comes from the legs that are in contact with the ground. The manipulator of the ROV system considered in this section does not generally come in contact with the ground. In addition, the ROV is assumed to be massive compared to the manipulator, or it is operating within a tight servo loop to maintain its desired motion. In either or both of these cases, the motion of the manipulator has a negligible effect on the motion of the ROV, and its dynamics can be decoupled from the manipulator. Therefore, its acceleration, $\mathbf{a}_0$, is assumed known when simulating the manipulator. This approach has two advantages: its simplifies the conceptual framework especially for the development of control systems, and lowers the computational requirements of the simulation.

In addition to the omission of the linear system solution to obtain $\mathbf{a}'_0$, the most significant reductions in the amount of computation come from the fact that $\boldsymbol{\beta}_0^H$, $\mathbf{I}_0^*$, and $\boldsymbol{\beta}_0^*$ are no longer needed. As a direct result, $\mathbf{N}_1$ also does not need to be computed. Finally, only the third elements (in the case of a revolute joint) of $\boldsymbol{\beta}_1^*$ and hence $\boldsymbol{\beta}_1^H$ are needed which requires [40M, 39A] and [81M, 55A], respectively. The computational requirements of the resulting algorithm are shown in Table 19 under the column labeled "Decoupled Dynamics." The total computational requirements at the bottom of this table show that the amount of computation using the decoupled dynamics approach is reduced by [397M, 360A] from the Full Dynamics approach discussed in the first part of this section. For the *Tiburon*/Schilling system this results in [1997M, 1733A] which is over 15% less than the Full Dynamics approach, and nearly 25% less than the general multiple chain algorithm. Before this method is used, additional experimentation to measure the errors that this simplification incurs should be carried out, to determine the efficacy of this approach.

7.6 Summary and Conclusions

In this chapter, an efficient dynamics algorithm has been developed for simulation of a URV system with multiple serial chains. The resulting general algorithm is capable of simulating both multilegged underwater vehicles, like the Aquarobot hexapod, and ROV/manipulator systems, like the Monterey Bay Aquarium Research Institute's ROV, *Tiburon*, with a Shilling manipulator. The AB approach to computing the dynamics for these systems has been shown to be most efficient, so it was used as the basis for this development. Using the efficient transformations from Appendix A, a more efficient algorithm for land-based robotic systems was developed first which required $[(224mN + 59m + 99)\text{M}, (205mN + 63m + 68)\text{A}]$ for an m-chain system with N revolute joints each.

A significant goal of this chapter was to efficiently incorporate the hydrodynamic effects presented in the previous chapter into this algorithm. After careful derivation,

Table 19: Computational requirements of the single chain URV simulation algorithm.

	Full Dynamics		Decoupled Dynamics	
	$\times$	$+$	$\times$	$+$
$\boldsymbol{v}_0^r$	12	9	12	9
${}^0\boldsymbol{a}_g$	4	—	4	—
${}^0\boldsymbol{a}_{f-g}$	12	7	12	9
$\boldsymbol{\beta}_0^H$	160	125	—	—
$\mathbf{v}_i^r$	$20N-12$	$13N-8$	$20N-12$	$13N-8$
${}^i\boldsymbol{a}_{f-g}$	$8N-4$	$4N-2$	$8N-4$	$4N-2$
$\boldsymbol{\zeta}_i$	$22N-20$	$10N-10$	$22N-20$	$10N-10$
$\boldsymbol{\beta}_i^H$	$160N$	$125N$	$160N-79$	$125N-70$
$\mathbf{n}_i$	—	—	—	—
$(m_i^*)^{-1}$	$N-1$	—	$N-1$	—
$\mathbf{n}_i(m_i^*)^{-1}$	$5N-5$	—	$5N-5$	—
$\mathbf{N}_i$	$15N-15$	$15N-15$	$15N-30$	$15N-30$
τ_i^*	—	N	—	N
$\mathbf{I}_{i-1}^*$	$70N-48$	$92N-52$	$70N-70$	$92N-92$
$\boldsymbol{\beta}_{i-1}^*$	$50N-27$	$49N-23$	$50N-60$	$49N-59$
$\mathbf{a}_0'$	86	71	—	—
$\mathbf{a}_0$	—	3	—	—
$\bar{\mathbf{a}}_i$	$20N-12$	$17N-11$	$20N-12$	$17N-11$
$\ddot{q}_i$	$6N$	$6N$	$6N$	$6N$
$\mathbf{a}_i'$	—	$N-1$	—	$N-1$
Total:	$377N+130$	$333N+93$	$377N-265$	$333N-265$
$(N=6)$	(2392)	(2091)	(1997)	(1733)

the computational requirement for the resulting underwater AB/DTS algorithm is $[(377mN + 83m + 256)\text{M}, (333mN + 77m + 202)\text{A}]$. For the Aquarobot this corresponds to [7540M, 6658A], and for *Tiburon* it is [2601M, 2277A]. Compared to the land-based algorithm, hydrodynamics increases the computation between 65% and 70% depending on the topology.

An advantage of this algorithm is that it is general enough to compute the dynamics of arbitrary mobile base, multiple chain systems. This is a disadvantage, however, when the system consists of a single serial chain such as an ROV with a single manipulator. With more efficient kinematic modeling, an algorithm specifically for single chain systems was also developed. The resulting algorithm requires $[(377N + 130)\text{M}, (333N + 93)\text{A}]$. For the *Tiburon* ROV with a Schilling manipulator ($N = 6$), this translates to [2392M, 2091A] which is 8% less computation. Often, another characteristic of these systems is that the ROV is massive compared to the manipulator, or under very tight servo control. Under either or both of these conditions, a simplified dynamics algorithm has also been developed which decouples the chain dynamics from the mobile base. The resulting algorithm requires $[(377N - 265)\text{M}, (333N - 265)\text{A}]$. For the *Tiburon*/manipulator system this corresponds to [1997M, 1733A] which as nearly 25% less than the general multiple chain algorithm.

Although hydrodynamics introduces a significant amount of computation of the dynamics for these systems, it is nearly low enough to enable real-time simulation on reasonably priced workstations. The goal of the next chapter is to use object oriented design and programming to obtain an efficient implementation of the general mobile base, multiple chain algorithm. It is implemented on an SGI workstation with a MIPS R4400 processor, and the computation time for the dynamics is determined in order to evaluate this statement.