Forward Dynamics with Spatial Vectors

1 The Dynamics Equation

The motion of any dynamic system can be described with the following equation:

$$\mathbf{Q} = H(\mathbf{q})\ \ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \tag{1}$$

where

$$\begin{array}{rcl}
\mathbf{Q} & = & \text{generalized forces} \\
\mathbf{q} & = & \text{generalized positions} \\
\dot{\mathbf{q}} & = & \text{generalized velocities} \\
\ddot{\mathbf{q}} & = & \text{generalized accelerations} \\
H(\mathbf{q}) & = & \text{mass / inertia coefficients} \\
\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) & = & \text{gravity and velocity contributions to } \mathbf{Q} \text{ (e.g. centrifugal and coriolis forces)}
\end{array}$$

For a single rigid body, Equation 1 becomes the familiar Newton-Euler dynamic equations:

$$\begin{bmatrix} \mathbf{f} \\ \tau \end{bmatrix} = \begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{x}} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} -m\mathbf{g} \\ \omega \times I\omega \end{bmatrix} \tag{2}$$

where

$$\begin{array}{rcl}
\mathbf{f} & = & \text{force, a vector in } \Re^3 \\
\tau & = & \text{torque, a vector in } \Re^3 \\
m & = & \text{mass of the body} \\
M & = & \text{mass matrix, scalar } m \text{ times the 3x3 identity matrix} \\
I & = & \text{inertia tensor of the body, a 3x3 matrix} \\
\mathbf{g} & = & \text{acceleration due to gravity, a vector in } \Re^3 \\
\ddot{\mathbf{x}} & = & \text{linear acceleration of the body, a vector in } \Re^3 \\
\omega & = & \text{angular velocity of the body, a vector in } \Re^3 \\
\dot{\omega} & = & \text{angular acceleration of the body, a vector in } \Re^3
\end{array}$$

In other words, for this example,

$$\mathbf{Q} = \begin{bmatrix} \mathbf{f} \\ \tau \end{bmatrix}, \quad H(\mathbf{q}) = \begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix}, \quad \ddot{\mathbf{q}} = \begin{bmatrix} \ddot{\mathbf{x}} \\ \dot{\omega} \end{bmatrix}, \quad \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} -m\mathbf{g} \\ \omega \times I\omega \end{bmatrix} \tag{3}$$

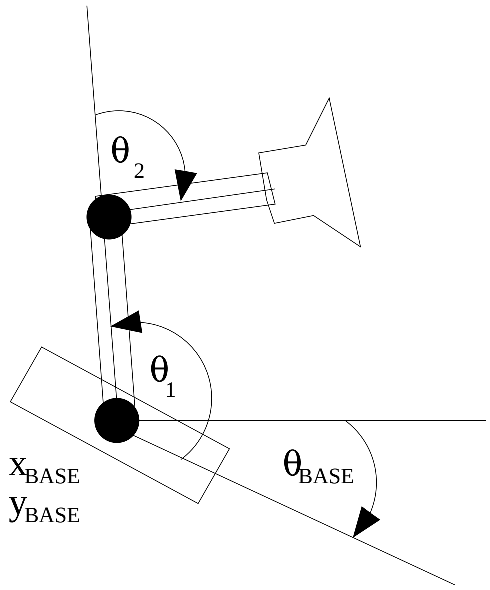

Figure 1: Generalized positions for a 2D jumping lamp.

An articulated body such as an animated character or robot will have a more complex set of equations. Setting up the dynamics of such a system starts with defining the generalized position terms $\mathbf{q}$ that will specify the configuration of the system. For a 2D jumping lamp with two joints (Figure 1), the pose of the lamp is completely determined by parameters $\mathbf{q}$ below:

$$\mathbf{q} = \begin{bmatrix} x_{BASE} \\ y_{BASE} \\ \theta_{BASE} \\ \theta_1 \\ \theta_2 \end{bmatrix} \tag{4}$$

There is a generalized force associated with each of the generalized position terms. A generalized force does work to cause a change in its corresponding position term. For example, torques at the joints are applied to cause changes in the joint angles at those joints. In this example, we have:

$$\mathbf{Q} = \begin{bmatrix} f_{BASE,x} \\ f_{BASE,y} \\ \tau_{BASE} \\ \tau_1 \\ \tau_2 \end{bmatrix} \tag{5}$$

For this system of linked bodies, matrix $H(\mathbf{q})$ and vector $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ are much more complex and laborious to calculate than the equivalent terms for a rigid body. Our goal is to compute these terms without ever having to look at them.

1.1 Forward and Inverse Dynamics

Given the dynamics equation, Equation 1, the forward and inverse dynamics problems are posed as follows:

- **Forward Dynamics:** Solve for accelerations $\ddot{\mathbf{q}}$ given forces Q, positions $\mathbf{q}$, and velocities $\dot{\mathbf{q}}$.
- **Inverse Dynamics:** Solve for forces Q given positions $\mathbf{q}$, velocities $\dot{\mathbf{q}}$, and accelerations $\ddot{\mathbf{q}}$.

Forward dynamics is used for simulation, e.g. to simulate a tablecloth being draped over a table or to simulate the motion of Luxo given joint torque inputs provided by the user. Inverse dynamics is used in robotics when the desired motion of the robot is known and joint torques must be computed to create that motion. Inverse dynamics is less directly useful in animation than in robotics, because an animated character can be moved along any desired path without knowing the required torques. Joint torques can be useful, however, for ensuring that motion falls within joint torque (or muscle force) limits. Counterintuitively, inverse dynamics is easier to compute than forward dynamics, and we will be using it in the forward dynamics computation.

2 Computing Forward Dynamics using Inverse Dynamics

For forward dynamics, we rewrite Equation 1 as follows:

$$H(\mathbf{q})\;\ddot{\mathbf{q}} = \mathbf{Q} - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \tag{6}$$

Q, $\mathbf{q}$, and $\dot{\mathbf{q}}$ are given. Once matrices $H(\mathbf{q})$ and $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ are determined, Equation 6 is a straightforward matrix equation that can be solved for $\ddot{\mathbf{q}}$.

The trick is to compute matrices H and $\mathbf{C}$ as efficiently and painlessly as possible. One elegant way to compute these matrices involves making calls to an inverse dynamics routine. Assume we have a function to compute inverse dynamics as follows:

$$InverseDynamics(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \tag{7}$$

The function *InverseDynamics* takes generalized positions, velocities, and accelerations and returns generalized forces. Referring to Equation 1, we see that:

$$InverseDynamics(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) = H(\mathbf{q})\;\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \tag{8}$$

We do not know $\ddot{\mathbf{q}}$, but let's assume it is zero. In this case (from Equation 8):

$$InverseDynamics(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{0}) = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \tag{9}$$

The H matrix disappears because it has been multiplied by a zero vector. Thus, calling $InverseDynamics$ with $\ddot{\mathbf{q}}$ set to zero gives us the $\mathbf{C}$ matrix directly.

Now, suppose that $\ddot{\mathbf{q}} = \delta_i$, where δ_i is a vector having the *ith* element one and all other elements zero. If we call inverse dynamics with this value of $\ddot{\mathbf{q}}$, we get:

$$InverseDynamics(\mathbf{q}, \dot{\mathbf{q}}, \delta_i) = H_i(\mathbf{q}) + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \tag{10}$$

where $H_i(\mathbf{q})$ is the ith column of the H matrix. Multiplying H by δ_i selects the ith column of H and ignores all other columns. This means we can solve for the H matrix column by column as follows:

$$H_i(\mathbf{q}) = InverseDynamics(\mathbf{q}, \dot{\mathbf{q}}, \delta_i) - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \tag{11}$$

The computation of $\mathbf{C}$ and H gives us everything we need to solve for $\ddot{\mathbf{q}}$.

In summary, the forward dynamics computation requires the following steps:

1. Use Expression 9 to find the $\mathbf{C}$ vector, the vector of gravity and velocity contributions to generalized force.
2. Use Expression 11 for each column i to fill in the H matrix, the matrix of mass and inertia terms.
3. Solve Equation 6 for accelerations $\ddot{\mathbf{q}}$.

All that remains is to describe the $InverseDynamics$ function.

3 Spatial Vector Inverse Dynamics

Inverse dynamics returns a vector of generalized forces $\mathbf{Q}$ that could have produced a given motion:

$$\mathbf{Q} = InverseDynamics(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \tag{12}$$

The inverse dynamics calculation can be written very compactly in recursive form as follows. The notation follows Featherstone.[1]

$$\begin{aligned}
\hat{v}_i &= {}_i\hat{X}_{i-1}\hat{v}_{i-1} + \hat{s}_i^{'}\dot{q}_i && (13)\\
\hat{a}_i &= {}_i\hat{X}_{i-1}\hat{a}_{i-1} + \hat{v}_i\hat{\times}\hat{s}_i^{'}\dot{q}_i + \hat{s}_i^{'}\ddot{q}_i && (14)\\
\hat{f}_i &= {}_i\hat{X}_{i+1}\hat{f}_{i+1} + \hat{I}_i^{'}\hat{a}_i + \hat{v}_i\hat{\times}\hat{I}_i^{'}\hat{v}_i - {}_i\hat{X}_{-1}\hat{f}_i^x && (15)\\
Q_i &= \hat{s}_i^{'S}\hat{f}_i && (16)
\end{aligned}$$

Appendices A and B define the terms and operators used in these equations. Note that the "hat" symbol indicates that a term is a **spatial vector**. It *does not* refer to a unit vector in $\Re^3$. Frame -1 will refer to the world frame, and so ${}_i\hat{X}_{-1}$ transforms external forces from the world frame to frame i.

At a high level, the algorithm works by accumulating velocities and accelerations outward toward the last joint (joint n). It then uses the velocity and acceleration values to calculate forces in the reverse direction, working from joint n inward toward the world frame, joint -1. The last equation simply takes the forces applied at a joint and projects them onto the relevant axis. For example, if we have a rotary joint with local axis in the z direction, only torques applied about the z-axis are of interest. (For a robot, these would be the only torques the robot's motors could generate.)

[1] R. Featherstone, "Robot Dynamics Algorithms," Kluwer Academic Publishers, 1987.

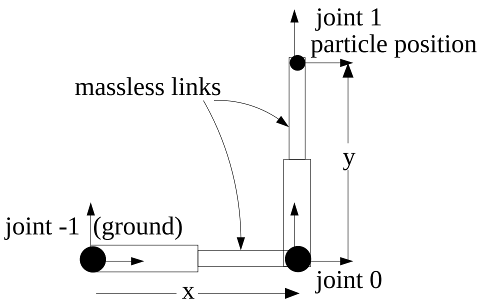

Figure 2: A particle can be thought of as being fixed to the ground through two massless prismatic joints, one that moves the particle in the x-direction and one that moves the particle in the y-direction.

A simple example follows in the next section. The example assumes you are familiar with spatial vectors and the terms in the inverse dynamics equations, so take some time to review Appendices A and B before working through the example.

3.1 Example: A Particle

This section goes through the complete inverse dynamics for a particle. (You should definitely use this derivation to debug your project.)

3.1.1 Setup

To set up the problem, we define generalized positions, which are just the x and y positions of the particle, and generalized forces, which are just forces in those directions:

$$\mathbf{q} = \begin{bmatrix} \mathrm{x} \\ \mathrm{y} \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} f_x \\ f_y \end{bmatrix} \tag{17}$$

The equations of motion are formulated for a character or robot fixed to the ground. To model a free particle, we create two imaginary joints – one that moves the particle in the x direction and another that moves the particle in the y direction (Figure 2). Because these are prismatic (translational) joints, their spatial vectors are formed as follows. (See Appendix A for details.) The local coordinate frames for the joints are shown in Figure 2.

$$\hat{s}_0^{'} = \begin{bmatrix} \mathbf{0} \\ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \end{bmatrix}, \quad \hat{s}_1^{'} = \begin{bmatrix} \mathbf{0} \\ \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \end{bmatrix} \tag{18}$$

Parameters $\hat{s}_0^{'}$ and $\hat{s}_1^{'}$ are 6x1 spatial vectors, and $\mathbf{0}$ is shorthand for a zero matrix of appropriate size (in this case, $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$). The first spatial vector represents linear motion in the x direction and the second represents linear motion in the y direction.

We will need the transformation matrices between joints. Here are the matrices from children to parents:

$$ {}_{-1}\hat{X}_0 = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -q_0 \\ 0 & q_0 & 0 \end{bmatrix} & \mathbf{1} \end{bmatrix}, \quad {}_0\hat{X}_1 = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \begin{bmatrix} 0 & 0 & q_1 \\ 0 & 0 & 0 \\ -q_1 & 0 & 0 \end{bmatrix} & \mathbf{1} \end{bmatrix} \tag{19} $$

Note that in these expressions, $\mathbf{1}$ is the 3x3 identity matrix, and $\mathbf{0}$ is a 3x3 matrix of zeroes. Referring to Appendix A, note that ${}_{-1}\hat{X}_0$ is a transformation matrix for a translation by q_0 in the x direction. In other words, moving from frame 0 to frame -1 means adding a translation of q_0 in the x direction, which does reflect the situation shown in Figure 2.

The next step is to define inertias. From joint 0 to joint 1, there is only a massless linkage, so $\hat{I}_0'$ is 0. Joint 1 pushes the particle directly, so $\hat{I}_1'$ will have a non-zero mass element:

$$ \hat{I}_0' = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \hat{I}_1' = \begin{bmatrix} \mathbf{0} & M \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{20} $$

$\hat{I}$ is a 6x6 matrix, $\mathbf{0}$ implies a 3x3 matrix of zeroes, and M is a 3x3 matrix with scalar particle mass m on the diagonal.

Finally, we define any external forces. Generalized force Q incorporates any "rocket" or contact forces that may be applied to the particle, and the only other forces are due to gravity. The easiest way to deal with gravity is to place the particle in an accelerating frame:

$$ \hat{v}_{-1} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \quad \hat{a}_{-1} = \begin{bmatrix} \mathbf{0} \\ \begin{bmatrix} 0 \\ g \\ 0 \end{bmatrix} \end{bmatrix} \tag{21} $$

where g is approximately $9.8m/s^2$. In other words, having the world accelerate upward is just like having the world sit stationary while gravity pulls down on luxo. Now that gravity has been dealt with, we conclude the problem setup by setting all external forces to zero:

$$ \hat{f}_0^x = \hat{f}_1^x = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{22} $$

3.1.2 Inverse Dynamics

Working through the inverse dynamics symbolically for a particle can provide some insight into the working of the dynamics equations. This section provides partial results, but leaves the detailed working out of equations up to you. We start with velocities:

$$ \hat{v}_i =_i \hat{X}_{i-1}\hat{v}_{i-1} + \hat{s}_i'\dot{q}_i \tag{23} $$

$$\hat{v}_{-1} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \quad \hat{v}_0 = \begin{bmatrix} \mathbf{0} \\ \begin{bmatrix} \dot{q}_0 \\ 0 \\ 0 \end{bmatrix} \end{bmatrix}, \quad \hat{v}_1 = \begin{bmatrix} \mathbf{0} \\ \begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ 0 \end{bmatrix} \end{bmatrix} \tag{24}$$

Accelerations:

$$\hat{a}_i = {}_i\hat{X}_{i-1}\hat{a}_{i-1} + \hat{v}_i \hat{\times} \hat{s}_i' \dot{q}_i + \hat{s}_i' \ddot{q}_i \tag{25}$$

$$\hat{a}_{-1} = \begin{bmatrix} \mathbf{0} \\ \begin{bmatrix} 0 \\ g \\ 0 \end{bmatrix} \end{bmatrix}, \quad \hat{a}_0 = \begin{bmatrix} \mathbf{0} \\ \begin{bmatrix} \ddot{q}_0 \\ g \\ 0 \end{bmatrix} \end{bmatrix}, \quad \hat{a}_1 = \begin{bmatrix} \mathbf{0} \\ \begin{bmatrix} \ddot{q}_0 \\ \ddot{q}_1 + g \\ 0 \end{bmatrix} \end{bmatrix} \tag{26}$$

Forces:

$$\hat{f}_i = {}_i\hat{X}_{i+1}\hat{f}_{i+1} + \hat{I}_i'\hat{a}_i + \hat{v}_i \hat{\times} \hat{I}_i' \hat{v}_i - {}_i\hat{X}_{-1}\hat{f}_i^x \tag{27}$$

$$\hat{f}_1 = \begin{bmatrix} \begin{bmatrix} m\ddot{q}_0 \\ m(\ddot{q}_1 + g) \\ 0 \end{bmatrix} \\ \mathbf{0} \end{bmatrix}, \quad \hat{f}_0 = \begin{bmatrix} \begin{bmatrix} m\ddot{q}_0 \\ m(\ddot{q}_1 + g) \\ 0 \end{bmatrix} \\ \begin{bmatrix} 0 \\ 0 \\ -mq_1\ddot{q}_0 \end{bmatrix} \end{bmatrix}, \quad \hat{f}_{-1} = \begin{bmatrix} \begin{bmatrix} m\ddot{q}_0 \\ m(\ddot{q}_1 + g) \\ 0 \end{bmatrix} \\ \begin{bmatrix} 0 \\ 0 \\ -mq_1\ddot{q}_0 + mq_0(\ddot{q}_1 + g) \end{bmatrix} \end{bmatrix} \tag{28}$$

Generalized forces:

$$Q_i = \hat{s}_i'^S \hat{f}_i \tag{29}$$

$$Q_0 = m\ddot{q}_0, \quad Q_1 = m(\ddot{q}_1 + g) \tag{30}$$

After all that, we end up with Newton's second law of motion: $F = ma$! The benefit of this approach, however, is that it is not much more difficult to set up the dynamics of a complex 3D multi-body system.

4 Summary

In summary, the forward dynamics problem can be solved by repeated calls to an inverse dynamics routine. Inverse dynamics can be written compactly using a spatial vectors approach. To set up a new problem, the following terms must be defined for the system:

- Generalized positions $\mathbf{q}$.
- Generalized forces $\mathbf{Q}$.

The following must be initialized:

- Joint axes in each body's local frame $\hat{s}_i'$
- Spatial transforms ${}_i\hat{X}_{i+1}$

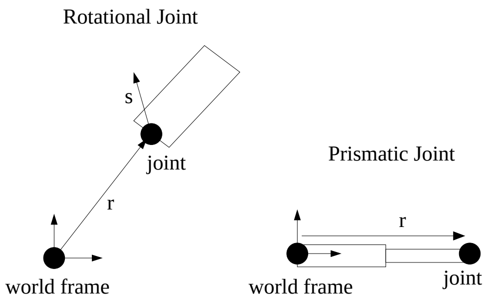

Figure 3: A spatial joint axis represents a single degree of freedom that can be rotational or translational (prismatic).

- Spatial inertias in each body's local frame $\hat{I}_i'$
- External forces $\hat{f}_i^x$
- World velocity and acceleration $\hat{v}_{-1}$, $\hat{a}_{-1}$

Spatial transforms, external forces, and generalized forces may change over the course of the simulation. All other terms should remain the same.

Appendix A: Definition of Terms

Terms used in this document are defined as follows:

$\hat{s}_i'$	spatial joint axis of link i in its local coordinate frame
$\hat{v}_i$	spatial velocity of link i
$\hat{a}_i$	spatial acceleration of link i
$\hat{f}_i^x$	spatial external forces applied to link i in world frame
$\hat{f}_i$	sum of spatial forces on link i
Q_i	scalar value of generalized force i
${}_k\hat{X}_j$	spatial transform from coordinate frame j to k
$\hat{I}_i'$	spatial inertia of link i in its local coordinate frame

The sections below outline the structure of spatial joint axes, spatial transforms, and spatial inertias.

4.1 Spatial Joint Axes

A spatial joint axis represents a single rotational or translational degree of freedom. A rotational joint is represented by its axis of rotation, $\mathbf{s}$, and by a term related to its position in space $\mathbf{r} \times \mathbf{s}$ (see Figure 3, left):

$$\hat{s} = \begin{bmatrix} \mathbf{s} \\ \mathbf{r} \times \mathbf{s} \end{bmatrix} \tag{31}$$

The position component $\mathbf{r} \times \mathbf{s}$ reflects the fact that the rotation at the joint occurs about a line. Mathematically, it does not matter where along the axis of rotation the actual joint lies; the effect of rotating about that axis is the same. (Term $\mathbf{r} \times \mathbf{s}$ will have the same value regardless of where $\mathbf{r}$ points to on the axis of joint rotation.)

For a prismatic joint as shown in Figure 3 (right), there is no axis of rotation, only translational motion, and $\hat{s}$ is defined as follows:

$$\hat{s} = \begin{bmatrix} \mathbf{0} \\ \mathbf{r} \end{bmatrix} \tag{32}$$

4.2 Spatial Transforms

A spatial transform is a 6x6 matrix that transforms spatial quantities from one coordinate frame to another. We can think about how to construct a spatial transform by looking at the case of transforming the spatial joint axis of a rotary joint:

$$\hat{s} = \begin{bmatrix} \mathbf{s} \\ \mathbf{r} \times \mathbf{s} \end{bmatrix} \tag{33}$$

A pure rotation would simply rotate each of the vectors $\mathbf{s}$ and $\mathbf{r} \times \mathbf{s}$:

$$\hat{X}_{ROT} = \begin{bmatrix} R & \mathbf{0} \\ \mathbf{0} & R \end{bmatrix}, \quad \hat{X}_{ROT}\, \hat{s} = \begin{bmatrix} R\mathbf{s} \\ R(\mathbf{r} \times \mathbf{s}) \end{bmatrix} \tag{34}$$

A pure translation leaves rotation axis $\mathbf{s}$ unchanged, but must alter the $\mathbf{r} \times \mathbf{s}$ term. Specifically, for a translation by vector $\mathbf{t}$, the new spatial joint axis should be:

$$\hat{s} = \begin{bmatrix} \mathbf{s} \\ (\mathbf{r} + \mathbf{t}) \times \mathbf{s} \end{bmatrix} \tag{35}$$

A transformation matrix that achieves this result is:

$$\hat{X}_{TRANS} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{t}^* & \mathbf{1} \end{bmatrix}, \quad \hat{X}_{TRANS}\, \hat{s} = \begin{bmatrix} \mathbf{s} \\ \mathbf{t} \times \mathbf{s} + \mathbf{r} \times \mathbf{s} \end{bmatrix} \tag{36}$$

where $\mathbf{t}^*$ is the cross product matrix:

$$\mathbf{t}^* = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \tag{37}$$

Transformations can be combined with ordinary matrix multiplication, so a rotation followed by a translation would look like:

$$\hat{X}_{ROT+TRANS} = \begin{bmatrix} R & \mathbf{0} \\ \mathbf{t}^* R & R \end{bmatrix} \tag{38}$$

4.3 Spatial Inertias

Spatial inertias have the following form:

$$\hat{I} = \begin{bmatrix} M\mathbf{c}^{*T} & M \\ I^* + \mathbf{c}^* M \mathbf{c}^{*T} & \mathbf{c}^* M \end{bmatrix} \tag{39}$$

where I^* is the ordinary 3x3 inertia tensor of the link about its center of mass and $\mathbf{c}$ is the vector from the origin to the center of mass of the link. The lower left term of this matrix is the tensor form of the parallel axis theorem: inertia about an axis displaced from the center of mass is equal to inertia at the center of mass plus mass times the displacement squared: $I = I_{COM} + md^2$.

Appendix B: Spatial Operators

There are two spatial operators used in the inverse dynamics equations that behave differently from their ordinary vector counterparts: the spatial cross product $\hat{\times}$ and the spatial transpose ^{S}.

The spatial cross product works as follows:

$$\hat{a}\hat{\times}\hat{b} = \begin{bmatrix} \mathbf{a} \\ \mathbf{a}_o \end{bmatrix} \hat{\times} \begin{bmatrix} \mathbf{b} \\ \mathbf{b}_o \end{bmatrix} = \begin{bmatrix} \mathbf{a} \times \mathbf{b} \\ \mathbf{a} \times \mathbf{b}_o + \mathbf{a}_o \times \mathbf{b} \end{bmatrix} \tag{40}$$

where $\times$ is the ordinary vector cross product.

The spatial transpose works as follows:

$$\hat{a}^S = \begin{bmatrix} \mathbf{a} \\ \mathbf{a}_o \end{bmatrix}^S = [\mathbf{a}_0 \quad \mathbf{a}] \tag{41}$$

For a detailed discussion of these operators, see Featherstone.[2]

[2]R. Featherstone, "Robot Dynamics Algorithms," Kluwer Academic Publishers, 1987.