Quaternion-Based Analytical Inverse Dynamics for the Human Body

Andrew Du

December 31, 2021

Abstract

The human body provides unique challenges to study from a dynamical perspective, due to its mechanical complexity and the difficulty of obtaining measurements of internal dynamic quantities. Thus, it is essential to create models that both simplify analysis and account for important anatomical details, the two of which must necessarily be balanced into a sufficiently accurate-yet-manageable framework. A number of critical applications require accurate inverse dynamic models of the human body, including medical treatment and virtual simulation of human motion. A recent general technique was developed by Dumas et. al. that used a quaternion screw algebra to make computation of inverse dynamic quantities more practical and more efficient. In this paper, we adapt their technique to the case of human anatomy, integrating these computational improvements within a novel framework for modeling human musculature.

Contents

1 Introduction

Inverse dynamics is essential to understanding physics in the human body. Using kinematic measurements from video- and marker-based tracking technologies [6], inverse dynamics allows for the calculation of dynamical quantities as well: forces, torques, and tensions experiences by body components. Inverse dynamics can be used for various medical purposes, such as the design of physical therapy treatments and prosthetics, or the accurate replication of human locomotion in digital media and for humanoid robots. In the first case, a proper understanding of forces and torques throughout the human body can lead to an improved understanding of human health, by comparing the maximum load on certain parts of a patient's body to that of a healthy human [8]. In the second case, achieving realistic animation requires inverse dynamics to ensure that digital replication of human motion is in accordance with the forces and torques implied [5].

The use of analytical inverse dynamic methods over numerical ones is of great interest to various models. In general, analytical techniques will yield exact solutions while numerical methods can only result in approximated answers, whose accuracy depends on the number of times an iterative procedure is repeated. Thus, analytical methods present greater potential in terms of computational efficiency, though at present analytic solutions also involve more complicated equations that take far longer to solve than those associated with numerical approaches.

In addition to the drawbacks of numerical methods, the models that are described by such equations have a number of limitations. In particular, practical models often require non-physical assumptions about the human body, such as rigid segments and idealized pin joints [4]. In particular, a key condition of many existing models is that only adjacent segments may influence each other; consequently, the net torques and forces are confined to a single joint between segments. The traditional method then begins at the center of pressure and moves towards the joint of interest by considering each new segment one at a time. However, various components of the body violate this principle, such as biarticular muscles that allow non-adjacent segments (usually determined by skeletal structure) to act on one another [1]. Consequently, a reworked model is necessary to capture these nuanced dynamics, in addition to application of analytical methods.

Typically, Newton's method in Cartesian coordinates is favored over other methods for its ease of implementation, though integrating Lagrange's equations of motion in other coordinate systems and the Featherstone algorithm have also been applied for certain purposes—the latter option is often considered too restrictive for practical use alone, though using Lagrange's equations is promising as a way to simplify certain models, usually requiring fewer but more complex equations than the standard Newton–Euler implementation, and applying the Featherstone algorithm to only certain portions of larger models may be considered [7].

In this paper, we adapt a recent alternative technique applying screw algebra and quaternions to models of human limbs [3]. With the more natural framework of quaternions, we are able to tackle more realistic and complex models of the human arm without resorting to numerical methods. In particular, we provide a natural way to model muscular connections between anatomical components of the arm.

After a brief overview of the quaternion algebra, and an introduction to our quaternion-wrench framework, we discuss three models of increasing complexity for the human arm and provide implementation details for each within our framework.

The code to all models can be found at [2].

2 Quaternions in Inverse Dynamics

2.1 The Quaternion Algebra

Traditional methods of inverse dynamics typically use Euler angles or Cardan angles (also known as pitch, yaw, and roll) to define orientation in three dimensional space. In this framework, orientation is defined by a series of three rotations about different axes. With Euler or Cardan angles, each rotation, in addition to turning the object itself, rotates the axes of subsequent rotations. However, this sequence of rotations may lead to two axes coinciding, a phenomenon known as *gimbal lock*. For example, in the case of Cardan angles, a yaw about the y-axis by 90 degrees causes the z-axis and the x-axis to coincide. Once this occurs, the pitch and roll occur around the same axis, meaning not all orientations can be reached from this state. However, to resolve this, we may turn to quaternions, an alternate parametrization of three-dimensional orientations that avoids singularities such as gimbal lock at the cost of one additional coordinate.

Definition 2.1 (Quaternions). The algebra $\mathbb{H}$ of quaternions is the four-dimensional vector space of elements $a + bi + cj + dk$, with $a, b, c, d \in \mathbb{R}$, endowed with the following (noncommutative) multiplication:

$$i^2 = j^2 = k^2 = ijk = -1.$$

The norm of a quaternion $q = a+bi+cj+dk$ is defined as $\|q\| = \sqrt{a^2 + b^2 + c^2 + d^2}$. Its conjugate is defined as $q^* = a - bi - cj - dk$.

Of primary interest will be quaternions of norm 1, due to the connection with 3-dimensional rotation they possess. Unit quaternions are able to be changed to the form

$$q = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)(xi + yj + zk),$$

where $\langle x, y, z\rangle$ is a unit vector. The importance of this format becomes apparent when considering quaternion multiplication, where conjugating an orientation or position vector in $\mathbb{R}^3$ (considered as the subset $\text{span}\{i, j, k\} \subset \mathbb{H}$) by a unit quaternion applies a counterclockwise rotation of θ about the axis determined by vector $\langle x, y, z\rangle$.

Definition 2.2 (Quaternion Rotation). Given a unit quaternion $q = \cos\left(\frac{\theta}{2}\right)+\sin\left(\frac{\theta}{2}\right)(xi + yj + zk)$, we define the map $R_q : \mathbb{R}^3 \to \mathbb{R}^3$ as $R_q(v) = qvq^*$, where $\mathbb{R}^3$ is identified with $\text{span}\{i, j, k\} \subset \mathbb{H}$. This map defines a rotation of v by an angle θ about the axis determined by vector $\langle x, y, z\rangle$.

Quaternions present a number of unique advantages over other representations of orientation in three dimensional space, such as Euler and Cardan angles. The latter are typically favored for their using only three coordinates and seeming more natural conceptually. However, quaternions avoid the issue of gimbal lock, in which a degree of freedom is lost, preventing all orientations from being properly represented. Additionally, transformations between any two orientations can be achieved by a single rotation about some axis (being the cross product of the two orientation vectors), or a single quaternion, whereas Euler and Cardan angles may require up to three rotations, assuming gimbal lock has not already prevented such a transformation from being possible. Lastly, quaternion multiplication is of great use for the purposes of computational

efficiency [3], and as will be seen in the following section, leads to natural and easily implemented equations of motion.

2.2 Equations of Motion

The inverse dynamics of the human body are naturally accommodated by quaternions, given the large number of joints and distinct orientations of components. Each joint may be tracked through only its rotations, with its degrees of freedom determined by the degrees of freedom of the joint's unit quaternion, and the length of connecting segments. Moreover, quaternions can be used to easily find the linear and angular velocity and acceleration of objects as well as the attitude matrix for the inertia tensor. Following the discussion in [3], we use the equations relating the forces and torques at the ends of segments to each other and those converting attitude quaternions to attitude matrices.

Fix a segment i in our system. We let q_s be the scalar or first component of the attitude quaternion of segment i while $\mathbf{q}_v$ is similarly set as its vector component, or its latter three values. Let $\mathbf{E}_{3\times 3}$ be the 3 dimensional identity matrix. For a vector $\mathbf{v} = \langle x_1, x_2, x_3 \rangle$, we let its skew-symmetric matrix $\tilde{\mathbf{v}}$ be

$$\tilde{\mathbf{v}} = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix},$$

or equivalently

$$\tilde{\mathbf{v}} = \begin{bmatrix} \mathbf{v} \times \vec{\mathbf{i}} & \mathbf{v} \times \vec{\mathbf{j}} & \mathbf{v} \times \vec{\mathbf{k}} \end{bmatrix}.$$

In this setting, the attitude matrix corresponding to the attitude quaternion of segment i is as in [3]:

$$\mathbf{R}_i = \left[\|q\|^2 \mathbf{E}_{3\times 3} + 2\mathbf{q}_v \mathbf{q}_v^T + 2q_s \tilde{\mathbf{q}}_v \right]_i. \tag{1}$$

We define $\mathbf{c}_i$ to be the vector from the proximal (adjacent to segment $i+1$) end of the i-th segment to its center of mass, and $\mathbf{d}_i$ to be the vector from the proximal end of the i-th segment to its distal (adjacent to segment $i-1$) end—in other words, the vector describing the orientation and length of the segment itself. Let $\mathbf{I}_i$ be the inertia tensor of the i-th segment in the inertial coordinate system (ICS), found through multiplication by its corresponding attitude matrix found from (1) to the previously known principal inertia tensor of the same segment, and here, the kinematic quantities involved are denoted as found in the physics literature: $\mathbf{F}$ for force, $\mathbf{M}$ for moment, m for mass, $\mathbf{a}$ for linear acceleration, $\boldsymbol{\alpha}$ for angular acceleration, $\boldsymbol{\omega}$ for angular velocity, and $\mathbf{g}$ for gravitational acceleration. The equation for the net force and moment of adjacent joints on the proximal end of segment i is as in [3]:

$$\begin{bmatrix} \vec{\boldsymbol{F}}_i \\ \vec{\boldsymbol{M}}_i \end{bmatrix} = \begin{bmatrix} m_i \mathbf{E}_{3\times 3} & \mathbf{0}_{3\times 3} \\ m_i \tilde{\mathbf{c}}_i & \mathbf{I}_i \end{bmatrix} \begin{bmatrix} \vec{\boldsymbol{a}}_i - \vec{\boldsymbol{g}} \\ \vec{\boldsymbol{\alpha}}_i \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{3\times 1} \\ \vec{\boldsymbol{\omega}}_i \times \mathbf{I}_i \vec{\boldsymbol{\omega}}_i \end{bmatrix} + \begin{bmatrix} \mathbf{E}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \tilde{\mathbf{d}}_i & \mathbf{E}_{3\times 3} \end{bmatrix} \begin{bmatrix} \vec{\boldsymbol{F}}_{i-1} \\ \vec{\boldsymbol{M}}_{i-1} \end{bmatrix}. \tag{2}$$

Generally, the inverse dynamic method begins at the extremal segment, segment 1, farthest from the joint of interest. The distal end of segment 1 is typically the location of the known external force $\mathbf{F}_0$ and moment $\mathbf{M}_0$, such as from a held weight, when considering human limbs. Using attitude matrices calculated from (1) on previously known inertia tensors, combined with known values of kinematic quantities, we may move along the segments with equation (2) to obtain the desired joint torques and forces.

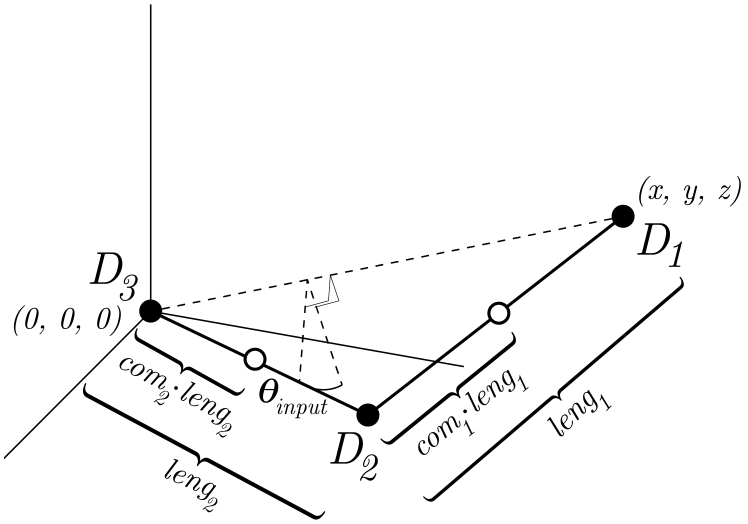

Figure 3.1.1: Two segment model of arm

3 Models

In this section we discuss the human arm as it experiences various forces and torques. We study models of increasing complexity in musculature and structure, applying equations with the previously discussed quaternion framework to obtain desired values. Here we focus on the torques and forces exerted on the shoulder and arm components from the action of lifting an object in the hand. The initial model includes two segments of no volume without any simulation of muscular force, while subsequent models add on, in turn, the biceps and triceps, and the hand. The same techniques and frameworks presented in this paper may be extended to other systems, as described in the discussion following these models.

3.1 A Static Arm with No Musculature Holding a Weight

The initial model (See Figure 3.1.1) treats the arm as two segments connected by a joint and does not consider movement of any kind. The code for this may be found at [2]. No musculature is considered at present, though we provide a framework into which muscles may be easily incorporated through the addition of pairs of forces originating from the endpoints of muscles.

The shoulder and end of the forearm (which is considered the hand and holds a point weight of known mass) are both in fixed positions at $(0, 0, 0)$ and (x_1, y_1, z_1), respectively. Thus, there is a single degree of freedom: the rotation of the arm about the vector $\overrightarrow{D_3D_1}$, which determines along with the predetermined lengths of the segments the position of D_2, the elbow. We define the rotation of the arm by the angle θ_{input} around $\overrightarrow{D_3D_1}$, defining the $\theta_{\text{input}} = 0$ orientation as the one with such that the elbow D_2 has minimal z coordinate value in the ICS—the edge cases of a directly upwards or downwards arm, should they even need to be considered, may be dealt with in this model by simply introducing a secondary condition of minimal x or y value.

We set the principal axes of each segment such that the x-axis connects its proximal end to its distal end, and such that the unit z vector in each SCS has maximal Cartesian z value in the ICS. Since our segments are stationary, these axes and thus the attitude quaternion for each segment are fixed.

Since the segments are one dimensional, we may treat the center of mass as a ratio of the overall length of the vector. This may be changed in more complex models that consider variable structure and volume.

As there is not yet a segment corresponding to the hand, we treat it as the distal end of the first segment at (x_1, y_1, z_1).

The code corresponding to this model (found at [2]) solves for the moment and force applied to the proximal end of each segment, or the shoulder and elbow, caused by forces and torques from the weight of the arm itself and an object held at D_1, the model's approximation for the hand.

The process begins with using the law of cosines to solve for the angles of the triangle formed by the endpoints of the two segments, found through the location of the hand at (x, y, z) and shoulder at the origin in combination with the lengths of the two segments, also known beforehand:

$$\cos(\theta) = \frac{x^2 + y^2 + z^2 - \|\overrightarrow{\boldsymbol{d}}_1\|^2 - \|\overrightarrow{\boldsymbol{d}}_2\|^2}{2\|\overrightarrow{\boldsymbol{d}}_1\|\|\overrightarrow{\boldsymbol{d}}_2\|}.$$

From here the location of the elbow, or D_2, begins at the zero degree location and the position of each arm segment is set by rotation by the input angle quaternion, whose axis is $\overrightarrow{D_3D_1}$ and therefore only affects D_2, leaving the positions of the shoulder and hand, as well as the segments' lengths and connection, intact. Afterwards, we may construct the attitude quaternions for each segment by starting with the previous segment (or $\overrightarrow{D_3D_1}$ in the case of the first segment) and multiplying it by the appropriate magnitude and quaternion with axis determined by the cross product of the previous and current segment and the previously calculated angle—in short, we set up the segments by finding each in terms of transformations applied to the previous, beginning with the proximal segment. Below, we let $\overrightarrow{\boldsymbol{d}}_{ix}$, $\overrightarrow{\boldsymbol{d}}_{iy}$, $\overrightarrow{\boldsymbol{d}}_{iz}$ be the respective x, y, and z-coordinates of vector $\overrightarrow{\boldsymbol{d}}_i$, $\times_{\mathbb{H}}$ denotes quaternion multiplication, and $U(\overrightarrow{\boldsymbol{v}}) = \frac{\overrightarrow{\boldsymbol{v}}}{\|\overrightarrow{\boldsymbol{v}}\|}$:

$$q_i = U\left(\begin{bmatrix} \sin^{-1}\left(\frac{\overrightarrow{\boldsymbol{d}}_{iz}}{\|\overrightarrow{\boldsymbol{d}}_i\|}\right) \\ \langle \overrightarrow{\boldsymbol{d}}_{ix}, \overrightarrow{\boldsymbol{d}}_{iy}, 0\rangle \times \overrightarrow{\boldsymbol{d}}_i \end{bmatrix} \times_{\mathbb{H}} \begin{bmatrix} \tan^{-1}\left(\frac{\overrightarrow{\boldsymbol{d}}_{iy}}{\overrightarrow{\boldsymbol{d}}_{ix}}\right) \\ 0 \\ 0 \\ 1 \end{bmatrix}\right).$$

With the attitude quaternions, we now find the attitude matrix for each segment using (1) and convert each inertia tensor in the SCS to its ICS equivalent with said matrix. Then, we apply a simplified version of (2) to find the net force and torque at D_2, the elbow, where $\overrightarrow{\boldsymbol{m_0 g}}$ is the weight of the held object as a vector $\langle 0, 0, -m_0 g\rangle$:

$$\begin{aligned}
\begin{bmatrix} \overrightarrow{\boldsymbol{F}}_1 \\ \overrightarrow{\boldsymbol{M}}_1 \end{bmatrix} &= \begin{bmatrix} \overrightarrow{\boldsymbol{m_0 g}} \\ \vec{0} \end{bmatrix}, \\
\begin{bmatrix} \overrightarrow{\boldsymbol{F}}_2 \\ \overrightarrow{\boldsymbol{M}}_2 \end{bmatrix} &= \begin{bmatrix} m_1\mathbf{E}_{3\times3} & \mathbf{0}_{3\times3} \\ m_1\tilde{\mathbf{c}}_1 & \mathbf{I}_1 \end{bmatrix} \begin{bmatrix} -\overrightarrow{\boldsymbol{g}} \\ \overrightarrow{\mathbf{0}} \end{bmatrix} + \begin{bmatrix} \mathbf{E}_{3\times3} & \mathbf{0}_{3\times3} \\ \tilde{\mathbf{d}}_1 & \mathbf{E}_{3\times3} \end{bmatrix} \begin{bmatrix} \overrightarrow{\boldsymbol{F}}_1 \\ \overrightarrow{\boldsymbol{M}}_1 \end{bmatrix}, \\
\begin{bmatrix} \overrightarrow{\boldsymbol{F}}_3 \\ \overrightarrow{\boldsymbol{M}}_3 \end{bmatrix} &= \begin{bmatrix} m_2\mathbf{E}_{3\times3} & \mathbf{0}_{3\times3} \\ m_2\tilde{\mathbf{c}}_2 & \mathbf{I}_2 \end{bmatrix} \begin{bmatrix} -\overrightarrow{\boldsymbol{g}} \\ \overrightarrow{\mathbf{0}} \end{bmatrix} + \begin{bmatrix} \mathbf{E}_{3\times3} & \mathbf{0}_{3\times3} \\ \tilde{\mathbf{d}}_2 & \mathbf{E}_{3\times3} \end{bmatrix} \begin{bmatrix} \overrightarrow{\boldsymbol{F}}_2 \\ \overrightarrow{\boldsymbol{M}}_2 \end{bmatrix}.
\end{aligned} \tag{3}$$

Since there is no movement in this model, the dynamic wrench is ignored entirely and the angular and linear acceleration in the first term on the right hand side are both zero. Note that this makes the inertia tensor irrelevant, though for the sake of preparation for adding motion we implement it in the algorithm.

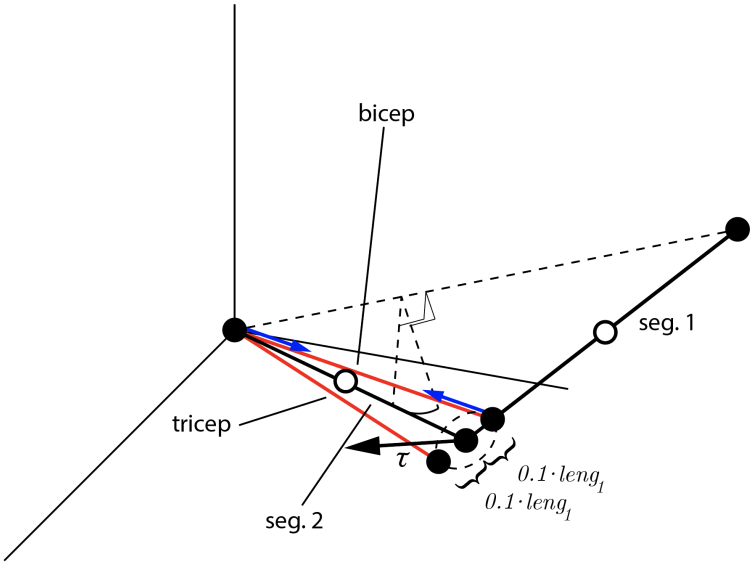

Figure 3.1.2: The triceps and biceps (red) are added and exert forces such that the moment on the elbow is coplanar with the segments and muscles.

3.2 A Static Arm with a Biceps and Triceps Holding a Weight

The following iteration of the model (3.1.2) introduces the first musculature in the form of a biceps and triceps. The code for this may be found at [2]. The first muscle is described by a tension force with endpoints at the shoulder D_3 (considered the bicep's proximal end) and some point (considered its distal end) along the forearm, or segment 1, whose position is described by a fraction of the segment vector $\vec{\boldsymbol{d}}_1$. We let this ratio be 0.9, though any other reasonable ratio suffices as well. Similarly, we model the triceps as a tension with endpoints at the shoulder and 1.1 along the length of the forearm, meaning this distal end is behind the upper arm. These two ratios, among other values, are stored in the `muscends` array, with four rows and one column per muscle (two here). Each column comes in the format `distal segment ratio`, `proximal segment ratio`, `distal segment index`, `proximal segment index`, referred to from here on as $r_d(j), r_p(j), n_d(j), n_p(j)$, respectively, for a muscle indexed j. So, our array for this model is

$$\begin{bmatrix} r_d(1) & r_d(2) \\ r_p(1) & r_p(2) \\ n_d(1) & n_d(2) \\ n_p(1) & n_p(2) \end{bmatrix} = \begin{bmatrix} 0.9 & 1.1 \\ 1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}.$$

Additionally, we create unit vectors $\vec{\boldsymbol{u}}_1, \vec{\boldsymbol{u}}_2$ representing the direction (distal to proximal) of the biceps and triceps, respectively.

The code is based off of the original program in the first model. After the forces and moments are found at a given joint without any muscle involvement (this step being identical to the first model), the muscle tensions are calculated to minimize the moment about the elbow and added force on the upper arm. Consequently, only one of the triceps or biceps will exert a force—these two muscles create torques on the elbow in exactly opposite directions because they and the arm segments are all coplanar, with the muscles

on opposite sides of the pivot point. The moment at the elbow should have no component perpendicular to the muscles or arm segments once the muscles are used.

Thus the code finds the vector normal to the plane formed with the muscle direction vectors as a basis and then the projection of the elbow's moment vector onto this first vector. We let $\overrightarrow{\boldsymbol{R}}_i$ be the portion of the moment at joint i that can be removed by muscle exertion. With our current musculature only $\overrightarrow{\boldsymbol{R}}_2$ is of interest, as the rest of the joint moments are unaffected:

$$\overrightarrow{\boldsymbol{R}}_2 = -\left(\frac{\overrightarrow{\boldsymbol{M}}_2 \cdot (\overrightarrow{\boldsymbol{u}}_2 \times \overrightarrow{\boldsymbol{u}}_1)}{\|\overrightarrow{\boldsymbol{u}}_2 \times \overrightarrow{\boldsymbol{u}}_1\|^2}\right)(\overrightarrow{\boldsymbol{u}}_2 \times \overrightarrow{\boldsymbol{u}}_1).$$

This projection is the moment that can be mitigated by the triceps and biceps combined, and we can now find the exact force needed for such a result. We now introduce the equation for $\overrightarrow{\boldsymbol{R}}_2$, in a form which can be easily generalized later. Let n_m be the number of muscles in the model, and μ_i the force exerted by muscle i. Then, summing the moment created by each force acting on the segment, found through the cross product of the lever arm and the force vector, we have:

$$\overrightarrow{\boldsymbol{R}}_2 = \sum_{\substack{1\le i\le n_m,\\ n_d(i)=2}} \left(\left((1-r_d(i))\,\overrightarrow{\boldsymbol{d}}_{n_d(i)}\right) \times \left(\mu_i \overrightarrow{\boldsymbol{u}}_i\right)\right). \tag{4}$$

If $\overrightarrow{\boldsymbol{R}}_2$ is in the same direction as the the cross product of the muscle direction vectors, then the moment on the elbow from the action of holding the object is causing flexion, meaning the triceps needs to be active to maintain a static position maintain a static position. In this case the magnitude of the muscular force, denoted μ, is

$$\mu_2 = \frac{\left\|\overrightarrow{\boldsymbol{R}}_2\right\|}{\left\|\overrightarrow{\boldsymbol{d}}_{n_d(2)} \cdot (1-r_d(2)) \times \overrightarrow{\boldsymbol{u}}_2\right\|}.$$

Otherwise, when the arm is naturally extending without muscle interference, the biceps is used and the force is of magnitude

$$\mu_1 = \frac{\left\|\overrightarrow{\boldsymbol{R}}_2\right\|}{\left\|\overrightarrow{\boldsymbol{d}}_{n_d(1)} \cdot (1-r_d(1)) \times \overrightarrow{\boldsymbol{u}}_1\right\|}.$$

In both cases, the other muscle's magnitude, μ_1 or μ_2 in the respective cases, will be 0 to minimize added force.

From here it is simply a matter of creating new muscle wrenches that are added to the calculation of (3). Rather than creating wrenches for each muscle, we note that (disregarding biarticular muscles for now) the first affected joint for each muscle will be the proximal end of the distal segment it connects to, namely $D_{n_d(j)+1}$ for muscle j. Then it is simpler to instead create muscle wrenches indexed along the segments as opposed to muscles, while iterating along the n_m muscle indices:

$$\begin{bmatrix}\overrightarrow{\boldsymbol{F}}_i\\ \overrightarrow{\boldsymbol{M}}_i\end{bmatrix}_m = \sum_{\substack{1\le j\le n_m,\\ n_d(j)+1=i}} \begin{bmatrix}\overrightarrow{\boldsymbol{u}}_j \cdot \mu_j\\ \overrightarrow{\boldsymbol{d}}_{n_d(j)} \cdot (1-r_d(j)) \times \overrightarrow{\boldsymbol{u}}_j \cdot \mu_j\end{bmatrix}. \tag{5}$$

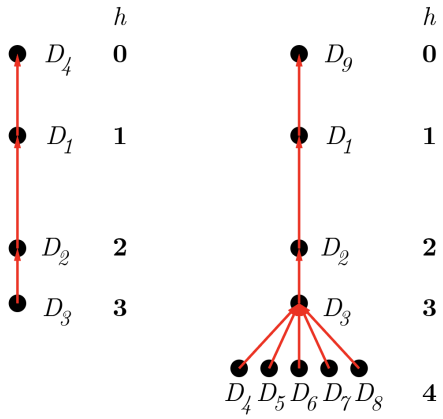

Figure 3.3.1: Indexing modified to fit the structure of a directed tree modeling the arm structure—this tree can be changed to fit any structure with no loops. The current model uses the left tree while the right tree is a possible tree for the addition of fingers, though the fingers would likely need to be split along knuckles and joints for accuracy. Calculation of wrenches now begins with the highest hierarchy value h, iterating downwards after exhausting all vertices of a given h.

Note that each $n_d(j)+1$ is not necessarily unique. The final step is to redo (3), with the modification of adding the muscle wrench:

$$\begin{bmatrix}\vec{F}_1\\ \vec{M}_1\end{bmatrix} = \begin{bmatrix}\overrightarrow{m_0 g}\\ \vec{0}\end{bmatrix},$$
$$\begin{bmatrix}\vec{F}_{i+1}\\ \vec{M}_{i+1}\end{bmatrix} = \begin{bmatrix}m_i\mathbf{E}_{3\times3} & \mathbf{0}_{3\times3}\\ m_i\tilde{\mathbf{c}}_i & \mathbf{I}_i\end{bmatrix}\begin{bmatrix}\vec{g}\\ \vec{0}\end{bmatrix} + \begin{bmatrix}\mathbf{E}_{3\times3} & \mathbf{0}_{3\times3}\\ \tilde{\mathbf{d}}_i & \mathbf{E}_{3\times3}\end{bmatrix}\begin{bmatrix}\vec{F}_i\\ \vec{M}_i\end{bmatrix} + \begin{bmatrix}\vec{F}_{i+1}\\ \vec{M}_{i+1}\end{bmatrix}_m,$$

for i from 1 to N (though in this model only $i=2$ has a relevant muscle wrench).

3.3 A Static Arm with Simple Upper Arm Musculature and a Hand

This time, we introduce a hand (without fingers yet) to the previously established model with a biceps and triceps. This is done with the addition of a third segment at the distal end of the forearm. Adding this hand creates a number of issues relating to inverse kinematic assumptions made by previous models, which we address in the following section.

3.3.1 Inverse Kinematics

Before describing major changes to the model due to the addition of the hand, it is important to note that the indexing system of the joints along the arm was changed significantly in preparation of the addition of fingers in the following models, which are not compatible with the sequential calculation of wrenches currently employed (3.3.1). First, due to segments only being added to the end of the arm, to avoid reindexing in later models the indices of existing joints and segments were partially flipped: the first segment is now the upper arm and indices increase from proximal to distal ends, instead. To handle the fingers, which would all contribute to the same wrench at the hand, the order of calculation was changed from linearly along indices to being defined by a tree. Each index i is assigned a single other index p_i towards which it points in a directed graph such that the resulting graph is a directed tree, ending in everything converging on the

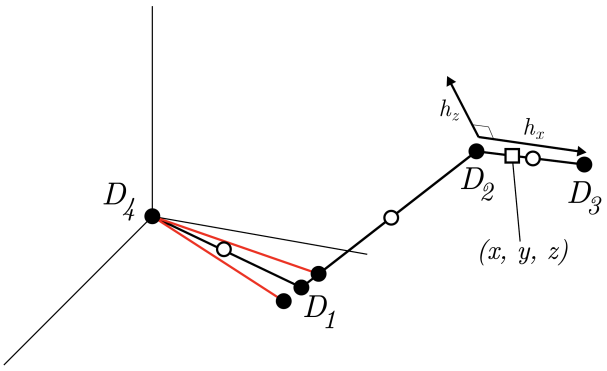

Figure 3.3.2: The third iteration of the model adds a hand as a third segment. Of note are the new input vectors $\vec{\mathbf{h}}_x$, $\vec{\mathbf{h}}_z$ determining the orientation of the x, z axes of the hand, and the point (x, y, z), where the object is held, which is no longer the same as the last distal end.

upper arm (indexed 1), which itself is assigned the index $n+1$ (still the shoulder) to point to. Each index's distance from the shoulder, in edges, is assigned to it as its hierarchy number h_i, forming a hierarchy in which the segments will always contribute to only the wrench of the segment their index points to. Thus, rather than iterating along indices for most purposes, we can now iterate along the assigned hierarchy number from highest to lowest. (The order in which results are calculated from segments of the same hierarchy number does not matter, as they cannot influence each other and the various wrenches are commutative under addition).

The first improvement to the model is the formalization of the weights (of which there is currently only one) to be implemented in a similar structure to muscles—they now have their own wrench type and the segment number, ratio defining position along said segment, and mass are all organized in matrices. The model used throughout this paper still only has one weight, but this implementation allows for easy addition and tracking of new weights along the arm. Similar to the definition of the `muscends` array, we now create a functional notation to describe the `weightsegs` matrix, which has a column vector for each muscle containing, from top to bottom, the index of the proximal end of the segment the weight directly affects, followed by the ratio along said segment, from the distal end, that the weight force is applied. Let $w_1(j), w_2(j)$ be the former and latter values, respectively, for muscle indexed j. Currently the only weight is at the hand, with $w_1(1) = 3, w_2(1) = 0.75$.

The addition of the hand gives a more natural way to begin the calculation of segment and endpoint locations. We let vectors $\overrightarrow{\boldsymbol{h}}_x$, $\overrightarrow{\boldsymbol{h}}_z$ describe the directions of the x- and z- axes of the hand in its SCS.

As with before we have x, y, z be the coordinates of the held object. From here we can calculate the position of the wrist and end of hand, D_2 and D_3 respectively:

$$D_2 = \begin{bmatrix} x \\ y \\ z \end{bmatrix} - (w_2(1) \cdot l_3)\overrightarrow{\boldsymbol{h}}_x,$$

$$D_3 = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + ((1 - w_2(1)) \cdot l_3)\overrightarrow{\boldsymbol{h}}_x.$$

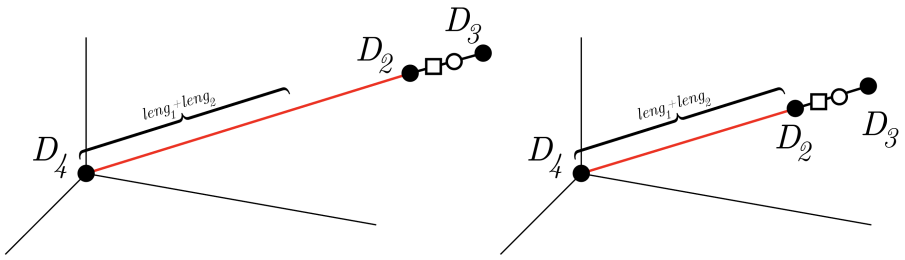

Figure 3.3.3: The two trivial cases for the positioning of the elbow. When the distance between D_4 and D_2 is too large, no solutions exist, while if the distance is exactly the sum of the lengths of the upper arm and forearm we have only one.

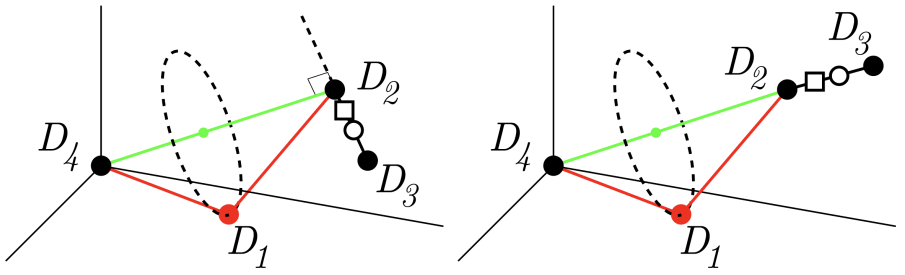

Figure 3.3.4: The first subcase, which itself contains two subcases. In these two cases, we cannot extend $\overline{D_2D_3}$ to find a point on the dashed circle (representing the locus of possible elbow points) to find the one that minimizes wrist bending (which in the following case is the unique point on the circle closest to the line), so we impose other restrictions, described below. The green line is the axis the circle is centered on and perpendicular to, while the red point and line segments are the parts of the arm affected by the position of the elbow—note that the hand is already fixed by user input, while the shoulder is fixed at the origin. Refer to figure 3.3.2 for a more detailed explanation of the model's structure as a whole.

With these two points, we proceed with the calculation of the rest of the points of the arm, the process of which is altered significantly by the hand.

The most critical assumption which now fails is that earlier iterations of code largely confine the model to a single plane, the 3-d orientation of which could be conveniently modified with only θ_{input} and its corresponding quaternion, without changing the position of the point of contact with the held object. The new segment adds multiple degrees of freedom to the possible orientations of segments that reach the held object. As opposed to requiring further user input, which would increase exponentially with segments added, the code takes in segment lengths, the position of the held object, and the orientation of the hand (defined by vectors for its x- and z- axes) as inputs, then determines the position of the elbow that minimizes bending at the wrist, measured by the angle of the shorter arc coplanar to the extension of the hand segment and the extension of the forearm line segment based on the proposed elbow position.

Note that the locus of possible elbow points is always empty, a point, or a circle, based on the value of the distance of D_2 from the origin in relation to the sum of the forearm and upper arm lengths l_1+l_2 (3.3.3). We determine which is the case for a given input, and proceed from there. If no points exist, the model has no solution, while having a single point means we only have one choice for the elbow—the interesting portion occurs when $l_1 + l_2 < \|D_2\|$. In this last case, there are two subcases. When considering the plane of the circle and the line formed by extension of the hand segment, we have two cases: the line intersects the plane at the center of the circle or is parallel to it (these two count as the same subcase due the code to handle them being identical), or the line intersects the plane at some point besides the circle's center.

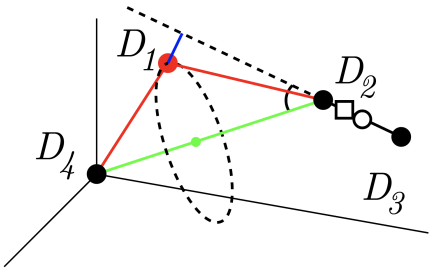

Figure 3.3.5: When there exists a point on the circle of possible elbow points with minimal distance from the extension of the hand segment (shown in blue), we set this to be the elbow point to minimize wrist bending.

In the first subcase (3.3.4), we choose a position such that the forearm is coplanar with the x- and z-axes of the hand (inputted by the user, with the x-axis along the length of the segment), of which there are two, modelling the tendency for the forearm's x- and z- axes to be coplanar with the direction of the fingers and palm. Between the two, there will always be one position of the elbow that results in an backwards rotation of the hand relative to the forearm and one that results in a forwards one; as before, based on the observed tendencies of the human arm we choose the position that results in this forward bend.

The other subcase (3.3.5) is easier to deal with, as the position of the elbow with minimal bending is unique; we find the point on the circle closest to the point of intersection between the line and plane, which can be done by adding an appropriately scaled vector pointing from the center of the circle towards the intersection point to the center of the circle.

3.3.2 Inverse Dynamics

The inverse dynamics of this model have changed from previous iterations due to the re-indexing, weight formalization, and change in location of the held object to along the hand. The wrenches of points with no connections, i.e. the fingers, are $\vec{\mathbf{0}}$ by default. However, it is probable that forces or moments will be explicitly coded to act on those points on a case by case basis, likely normal force from holding an object.

As mentioned in the previous section, the structure of the hand is now more accurately reflected in the order of calculation of wrenches. The code begins with `segcons`, a user inputted vector which contains at index i the index of the segment that the i-th connects to on its proximal end—these connections are the same for joints. The i-th element of this vector will be referred to as s_i. Vector `treehier` contains the aforementioned hierarchy value for each joint at its respective index, denoted h_i for joint i.

We now have, where n_w is the number of weights,

$$\begin{bmatrix} \vec{\boldsymbol{F}}_i \\ \vec{\boldsymbol{M}}_i \end{bmatrix}_w = \sum_{\substack{1 \le j \le n_w, \\ w_1(j)=i}} \begin{bmatrix} 0 \\ 0 \\ \overrightarrow{\boldsymbol{m_i g}} \\ (\vec{\boldsymbol{d}}_i \cdot w_2(j)) \times \overrightarrow{\boldsymbol{m_i g}} \end{bmatrix},$$

for the weight wrenches, while the muscle wrench calculation remains unchanged (5). Meanwhile, the joint wrench equations are modified to be

$$\begin{bmatrix}\overrightarrow{\boldsymbol{F}}_i\\ \overrightarrow{\boldsymbol{M}}_i\end{bmatrix} = \begin{bmatrix}\overrightarrow{\boldsymbol{F}}_i\\ \overrightarrow{\boldsymbol{M}}_i\end{bmatrix}_w + \begin{bmatrix}\overrightarrow{\boldsymbol{F}}_i\\ \overrightarrow{\boldsymbol{M}}_i\end{bmatrix}_m + \begin{bmatrix}m_i\mathbf{E}_{3\times3} & \mathbf{0}_{3\times3}\\ m_i\tilde{\mathbf{c}}_i & \mathbf{I}_i\end{bmatrix}\begin{bmatrix}-\overrightarrow{\boldsymbol{g}}\\ \overrightarrow{\mathbf{0}}\end{bmatrix} + \sum_{\substack{1\le j\le n,\\ p_j=i}}\left(\begin{bmatrix}\mathbf{E}_{3\times3} & \mathbf{0}_{3\times3}\\ \tilde{\mathbf{d}}_i & \mathbf{E}_{3\times3}\end{bmatrix}\begin{bmatrix}\overrightarrow{\boldsymbol{F}}_j\\ \overrightarrow{\boldsymbol{M}}_j\end{bmatrix}\right),$$

with wrenches of joints that do not have any connections in the distal direction being $\overrightarrow{\mathbf{0}}$ unless otherwise noted, as in the case of specific input forces or moments.

4 Discussion and Directions of Future Research

We now discuss several areas of improvement that may be addressed in models in the near future.

4.1 Improved Anatomy

Here we consider accuracy in representation of segments, as well as possible implementations of more complex systems of musculature, including biarticular muscles.

A known issue with the existing model is the accuracy of the attitude quaternions of the segments, more specifically that of the forearm in relation to the upper arm. The algorithm currently confines the axes of the SCS for these two segments such that the x and y of each are planar with the segments themselves, which frequently does not occur. However, this may be addressed with a solution similar to the one applied to the wrist, where certain criteria such as the angle between segments' SCS axes in the ICS are used to determine the natural rotation of the forearm in relation to the hand, and that of the upper arm in relation to the forearm.

The addition of more muscles is another area of interest. As mentioned previously in 4, the general equation solving for muscle exertion is

$$\overrightarrow{\boldsymbol{R}}_j = \sum_{\substack{1\le i\le n_m,\\ n_d(i)=j}}\left(\left((1-r_d(i))\,\overrightarrow{\boldsymbol{d}}_{n_d(i)}\right)\times\left(\mu_i\,\overrightarrow{\boldsymbol{u}}_i\right)\right),$$

though in the case of more muscles, case-specific constraints may be needed, such as reducing force on the wrist for medical treatment purposes.

As for biarticular muscles, the model does not consider the proximal end of muscles beyond for the purpose of the direction of the tension force, therefore they may be treated the same way as a normal muscle.

This model for muscles also gives a natural way to study forced movement of limbs by manually adding to muscle forces' magnitudes after solving for what forces are needed for equilibrium.

4.2 Fingers

Although the framework for their addition was set up in the third model, the implementation of the fingers themselves was avoided due to the vast amount of degrees of freedom they would add. However, in more specific cases where the shape of the held object is known, having a set of predetermined configurations may aid in this. Rather than predicting the position of all segments, which as noted before increases exponentially

in difficulty, testing a small number of expected positions of fingers may be sufficient. For example, for a hand holding a dumbbell, we may have presets for fingers holding the weight's rod part as a handle, as during a biceps curl, and another holding the plates as during an overhead triceps dip, though only with a single hand, as well as a third with the hand holding the object with the palm facing the wide face of the plate.

In this proposed implementation, we would simply preset attitude quaternions describing each segment's orientation in relation to the next one in the proximal direction, terminating in the segment just before the wrist. The algorithm would then test the possibility of each orientation and calculate the usually wrenches. Additional quantities of interest, such as the amount of bending done at a certain joint, may also be calculated with the attitude quaternions, and depending on the use case the viability of each hand preset may be judged with the desired criteria.

4.3 Analytical-Numerical Methods for Arms in Motion

At present the models have only considered the arm in a fixed position, and can only conveniently be extended to applied to other systems in static equilibrium. However, we may extend the model to study systems in motion, which [3] covers in the general wrench equation (2) while also providing a number of formulas for velocity and acceleration in relation to the attitude quaternions of segments. These analytical methods may then be used in conjunction with numerical ones and explicit integration techniques to arrive at a numerical solution.

5 Acknowledgements

I would like to thank my research mentor, David Darrow, for his guidance in this project, as well as Dr. Daniel Nolte for suggesting the topic of this paper. I am also equally grateful to Dr. Tanya Khovanova, Prof. Pavel Etingof, Dr. Slava Gerovitch for organizing this research opportunity as a part of the 2021 round of the MIT PRIMES-USA program.

References

[1] Daniel Cleather, Jon Goodwin, and Anthony Bull. An optimization approach to inverse dynamics provides insight as to the function of the biarticular muscles during vertical jumping. *Annals of biomedical engineering*, 39:147–60, 01 2011.

[2] A. Du and D. Darrow. Quaternion based inverse dynamic models. https://github.com/ddarrow90/inverse-dynamics, 2021.

[3] Raphael Dumas, Rachid Aissaoui, and Jacques de Guise. A 3d generic inverse dynamic method using wrench notation and quaternion algebra. *Computer methods in biomechanics and biomedical engineering*, 7:159–66, 06 2004.

[4] Herre Faber, Arthur Soest, and Dinant Kistemaker. Inverse dynamics of mechanical multibody systems: An improved algorithm that ensures consistency between kinematics and external forces. *PLOS ONE*, 13:e0204575, 09 2018.

[5] Hyeongseok Ko and N.I. Badler. Animating human locomotion with inverse dynamics. *IEEE Computer Graphics and Applications*, 16(2):50–59, 1996.

[6] Claudia Latella, Naveen Kuppuswamy, Francesco Romano, Silvio Traversaro, and Francesco Nori. Whole-body human inverse dynamics with distributed micro-accelerometers, gyros and force sensing. *Sensors*, 16(5):727, May 2016.

[7] Egbert Otten. Inverse and forward dynamics: Models of multi-body systems. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 358:1493–500, 10 2003.

[8] Joelly Mahnic de Toledo, Daniel Cury Ribeiro, and Jefferson Fagundes Loss. Análise por dinâmica inversa, um complemento da avaliação fisioterapêutica do ombro. *Fisioterapia e Pesquisa*, Sep 2009.