How to compute muscle moment arm using generalized coordinates

Michael Sherman, Ajay Seth, Scott Delp. *Rev. 0.2, 27 Oct 2010*

OpenSim provides to its biomechanics users a calculation called "muscle moment arm" which captures to some degree the leverage of a particular muscle with respect to a particular joint, while in a given configuration. For simple cases, this is the same as the conventional moment arm calculation in mechanical engineering. A straight-line muscle whose origin and insertion points connect two adjacent bodies connected by a pin joint is the simplest case. In practice, however, a muscle may span several joints (wrist, neck, spine) and will follow a contorted crossing over various curved bone surfaces. For these situations we need a careful definition of "muscle moment arm" that is analogous to the mechanical engineering concept and of use to biomedical researchers.

There are other related quantities that can easily be confused with moment arm, such as the amount of joint acceleration that a muscle can produce. That will be equivalent to moment arm in simple cases but confusingly different in more complex situations. Here we will give a precise definition of what we mean by "muscle moment arm" and, given that definition, how that quantity may be efficiently calculated. We will not attempt to address any deeper questions such as when you should be interested in moment arm or how you should use it.

Contents

Why this is a hard problem

Moment arm is such a simple concept in mechanics it is easy to assume it must be simple in biomechanics as well. Figure 1 shows a typical biomechanical exposition of the moment arm concept, commonly defined as “the distance from the muscle’s line of action to the joint’s center of rotation.”[1]

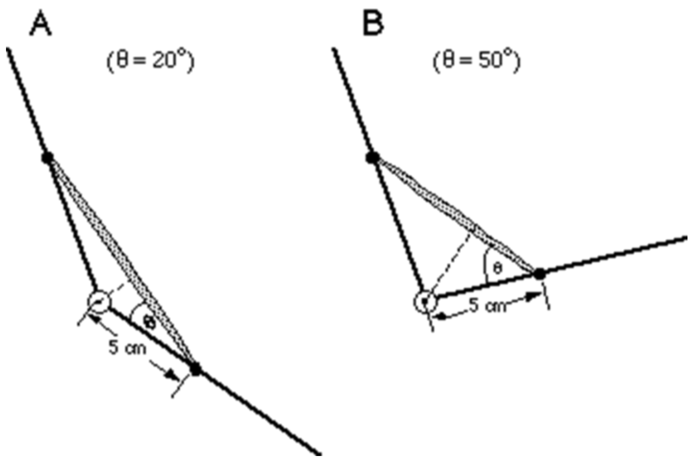

Figure 1: Oversimplified view of moment arm: “Although muscles produce linear forces, motions at joints are all rotary. … Mechanically, this is the distance from the muscle's line of action to the joint's center of rotation.”

However, biological systems are much more complex than typical engineered ones, and the mechanical models commonly used to study them have features not usually seen in machines. This leads to problems of definition, modeling, and implementation. What is the “line of action” if the muscle follows a curved path over the skeleton? What is the “center of rotation” when a muscle spans several joints? What if the spanned joint exhibits both rotational and translational motion? Figure 2 shows a finger model from ref. 2, which is representative of the more general situations for which moment arm must be defined and calculated. The figure shows the curved paths of three tendons, each crossing two joints. An associated muscle generates a scalar tension force s in its tendon; the moment arm r (with units of length) of that muscle about a “joint of interest” should characterize its effectiveness at generating a torque τ about that joint, such that $\tau=rs$. This is made more difficult by the fact that the “joint of interest” may not be any of the individual joints between bones, but can be a measurable quantity that is internally a composition of several joints. For example, Gonzalez, Buchanan, and Delp[3] modeled wrist flexion-extension kinematics as two coupled revolute joints with the total flexion angle divided evenly between the two. Then the moment arm of interest measures the effectiveness of the muscle at producing flexion.

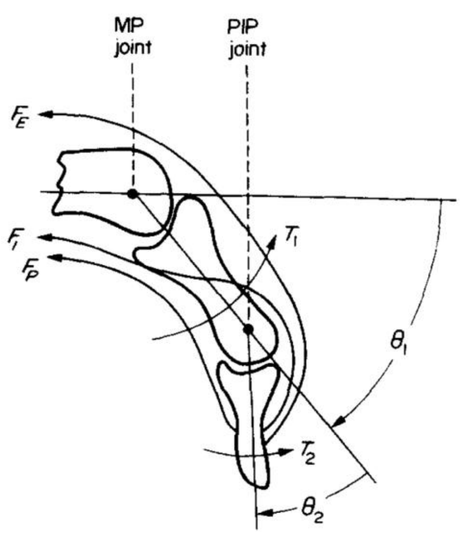

Figure 2: A more realistic model: a finger showing the curved paths of the extensor, profundus, and interosseous tendons (with tension forces F_E, F_I, F_P) crossing over two joints. A more general definition of moment arm is required.

As an example of implementation difficulty, consider that the desirable use of generalized (internal) coordinates in biomechanics may require coordinate choices that reflect the coupled rotations and translations generated by the complex geometry of joints.[4] That leads to generalized coordinates and generalized forces which cannot be interpreted directly in any familiar units, such as angles, lengths, forces, or moments.

Assumptions

A muscle is considered to connect body A to body B via a minimum-length curved path connecting an "origin" point on body A to an "insertion" point on body B, possibly passing frictionlessly over wrapping surfaces on these or other bodies. No particular connectivity is assumed for these bodies in the multibody model of the skeleton. For example, there might be a patella with wrap surface hanging off one of the bodies and driven from an arbitrary mobility.

Assumption 1: Although the path includes both muscle and tendon segments whose relative lengths vary dynamically, we are assuming that the total length l is just a kinematic quantity $l(q)$ (typically representing the shortest path) that can be calculated once the poses of all the bodies are known via specification of the generalized coordinates q. So for any given muscle we assume

$$l = l(q) \tag{1}$$

It is critical that any motion of points at which the muscle applies forces be modeled in a physically realizable manner so that muscle forces do the appropriate amount of work. (OpenSim's "moving muscle point" feature appears to violate this requirement.)

Assumption 2: Force generation by the muscle is completely characterized by a scalar tension $s \geq 0$ acting uniformly along the path, such that the set of spatial forces[*] F applied by the muscle in a given configuration is just a linear function of s:

$$F(s) = Ts \tag{2}$$

where T=$T(q)$ is the muscle's instantaneous "force transmission matrix". We expect that a muscle element can efficiently compute F given s, although T is not explicitly available.

Brief background

A few details are necessary for the derivation below.

[*] A spatial body force F^B is a pair of vectors: a moment applied to body B and a force applied at the body B origin; F here is a stacked vector of such spatial body forces.

1. Generalized speeds u are not always the same as time derivatives $\dot{q}$ of the generalized coordinates q, although they are closely related. This distinction will be important below because generalized forces are dual to the generalized speeds, but not to the generalized coordinate derivatives. In Simbody, the relationship is given by a block diagonal matrix N(q):

$$\dot{q} \triangleq \frac{dq}{dt} = Nu \tag{3}$$

2. There is a dual relationship between forces and velocities when measured in the same basis. The portion of a body B's spatial velocity V^B (at B's origin) due to generalized speed u_i is:

$$V_i^B = \frac{\partial V^B}{\partial u_i} u_i = J_i^B u_i \tag{4}$$

where Jacobian $J_i^B = J_i^B(q)$, and $V^B = \sum V_i^B$. The dual of this relationship relates a spatial force F^B applied at body B's origin to its contribution to the generalized force f_i acting at mobility u_i.

$$f_i = (J_i^B)^T F^B \tag{5}$$

For the dual relationship to hold, the generalized speeds and forces must be expressed in the same generalized basis.

3. The dynamic equations of motion for the multibody model representing our system are

$$M\dot{u} = f - (f_{\text{constraint}} + f_{\text{velocity}}) \tag{6}$$

where f is the generalized force equivalent of the applied forces, f_{velocity} is the gyroscopic and coriolis forces, and $f_{\text{constraint}}$ are the forces generated to enforce the constraints, all in generalized forces. The constraint forces are given by unknown multipliers:

$$f_{\text{constraint}} = G^T \lambda \tag{7}$$

We assume that the multibody dynamics system can calculate $\dot{u}, \lambda$ from the state and applied forces, and that given $\dot{u}, \lambda$ we can determine the applied forces from inverse dynamics.

Definition of "moment arm"

Muscle moment arm r is a measure of the effectiveness with which the contraction force of a given muscle can generate a torque about a "joint of interest", while in a given configuration q.

In general the joint of interest may reflect the complex combined effect of several internal components; we require only that there is a well-defined angular quantity $\theta=\theta(q)$ associated with the joint. Moment arm is a scalar quantity with units of length, and must depend only on the geometric properties of the system, that is, $r=r(q)$.

Moment arm is thus defined only for angles θ which determine muscle path length l kinematically, that is, we expect a virtual displacement $d\theta$ to produce a virtual length change dl that depends only on q, not on velocities, forces, or masses. All generalized coordinates q that can affect θ are thus assumed to be coupled; any that are not coupled explicitly by a constraint will be held constant during the moment arm calculation (that is, they are "coupled" with a coupling factor of 0; i.e., welded).

Given that requirement, we'll designate the muscle's moment arm with respect to a joint of interest as r_θ, and define it as follows:

$$r_\theta \triangleq \frac{\tau_\theta}{s} \tag{8}$$

where τ_θ is a scalar representing the effective torque acting about the angular coordinate θ, and $s>0$ is the scalar tension force generated by muscle activation.

When calculating moment arm, we expect that all constraints in the system are workless. That means that joints and wrapping surfaces are frictionless, and that all motions that can affect the muscle or joint are enforced using physically-valid constraint elements. With this requirement, the principle of virtual work allows us to write

$$s\,dl = \tau_\theta\,d\theta \tag{9}$$

where $l=l(q)$ is the muscle path length along which tension s is acting. Combining (9) with the moment arm definition (8) we have

$$r_\theta = \frac{dl}{d\theta} \tag{10}$$

Equation (10) provides a convenient method for calculating moment arm, but it must be emphasized that this is not a definition but a consequence of the assumption that all constraints are workless. Definition (8) is more general; however, we will only address in this paper moment arm calculations for which the assumptions behind equation (10) hold.

We note here that all the quantities we used in the definition above are ordinary physical quantities like angles, lengths, forces, and torques. In practice we build multibody models using generalized coordinates and corresponding generalized forces. It is important to realize that such quantities may be truly generalized, that is, they do not necessarily have physical units. Careful conversion is thus critical in order to use generalized coordinates to calculate physically-meaningful quantities like moment arm. This issue will be addressed below.

Specification of a "joint of interest"

An OpenSim user requesting a moment arm calculation will specify a muscle, and choose one of the available angular quantities for that muscle. Currently only a generalized coordinate subset is available for this purpose, so θ will always be one of the q's in the model. When the angle of interest is actually the sum of several coupled rotations, the coordinate associated with one of them (called the independent coordinate) is scaled so that it reads as the total angle rather than just the angle it controls directly. Coupler constraints are added separately to enforce the desired cooperative motion of the dependent coordinates. The algorithm below does not require this approach, but there must be a way to calculate θ from the q's and $\dot{\theta}$ from the $\dot{q}$'s.

Note that a muscle path may cross several independent coordinates, such as a hip and knee angle. When moment arm is calculated for one of those coordinates, the other is held rigid (meaning again that it is seen as coupled, but with a coupling factor of 0). Muscles crossing wrist, ankle, neck, and spine may be modeled with a single independent coordinate measuring the total angle, while several dependent coordinates are coupled to it.

Ways to calculate moment arm

Starting with the definition, there are a variety of ways to calculate moment arm differing in precision, implementation difficulty, and conceptual difficulty.

Finite differencing

We can calculate $r_\theta = dl/d\theta$ directly by finite differencing. That is, we can make a small perturbation $\Delta\theta$, satisfy all position constraints, update geometric calculations, and measure the resulting change Δl. The advantage of this method is that it directly implements the definition, and it is conceptually very simple. However, it has several drawbacks: it produces an approximate answer, and involves linearization difficulties due to the complex path geometry and the need to ensure satisfaction of the nonlinear position constraints. Also, because this is done at the position level it includes only holonomic constraints, and cannot account for nonholonomic constraints such as rolling.

Velocity-level kinematics

An easier and exact computation is available using velocities, since we have

$$r_\theta = \frac{dl}{d\theta} = \frac{dl/dt}{d\theta/dt} \equiv \frac{\dot{l}}{\dot{\theta}} \tag{11}$$

That is, if we can calculate $\dot{l}(\dot{\theta})$ then we need only enforce $\dot{\theta}=1$ (for example), satisfy all velocity constraints, then calculate $r_\theta = \dot{l}(1)$. This is probably the best way to calculate moment arm provided the operator $\dot{l}(\dot{\theta})$ is available. Unfortunately, it can be difficult to calculate $\dot{l}$ so we would like to find an alternative.

Partial velocity method

By assumption (1) above, we have $l = l(q)$ so

$$\dot{l} \triangleq \frac{dl}{dt} = \sum_i \frac{\partial l}{\partial q_i}\frac{\partial q_i}{\partial t} = P\dot{q} = PNu \tag{12}$$

with the last equality coming from equation (3). $P(q)$ is a row matrix whose i^{th} entry is the scalar $p_i = \partial l / \partial q_i$. If we can write

$$u = C\dot{\theta} \tag{13}$$

for some coupling matrix $C(q)$ (a column with entries c_i), then from equation (12) we have

$$\dot{l} = PNC\dot{\theta} \tag{14}$$

Then comparing (14) with (11) and noting that transposing a scalar doesn't change it, we have

$$r_\theta = PNC = C^T N^T P^T \tag{15}$$

If we had an explicit representation of P, this would be a very nice way to calculate r_θ. However, this would imply that we can calculate $\dot{l}(\dot{q})$ which we're assuming is difficult. Since generalized forces are dual to generalized speeds, we'll look at how to use forces instead of velocities to calculate r_θ.

Generalized force method

Simbody can map body spatial forces F to generalized forces f via an operator that calculates

$$f = J^T F \tag{16}$$

where J is the system Jacobian (partial velocity matrix) that maps generalized speeds to the body spatial velocities they produce. J just collects together the body Jacobians from equation (5).

Assumption (2) tells us how to calculate F from a given muscle tension scalar s, using the muscle's force transmission matrix T. Substituting (2) into (16):

$$f(s) = J^T T s \tag{17}$$

The column matrix $J^T T$ maps tension to generalized force; the dual problem mapping generalized speeds to $\dot{l}$ is then $\dot{l} = (J^T T)^T u$ (proof?). From equation (12) it follows that $J^T T = N^T P^T$. Substituting into equation (15) and using equation (17) gives

$$r_\theta = C^T N^T P^T = C^T (J^T T) = C^T f(s)/s \tag{18}$$

This gives us the algorithm we need for calculating moment arm without knowing how to calculate $\dot{l}$ directly:

1. Determine the coupling matrix *C* for the angular quantity of interest θ (see below)
2. Apply unit tension *s*=1 to the muscle of interest and map to muscle forces *F*(*s*) using operator (2)
3. Use the Simbody operator (16) to map muscle forces *F*(*s*) to generalized forces *f*(*s*) (see below)
4. Use equation (18) to compute $r_\theta = C^T f / s$ (see below)

Calculating the coupling matrix *C*

Set $\dot{\theta} = \dot{\theta}_0 = 1$, use Simbody's `project()` operator to find a least squares solution that satisfies the velocity constraints (this will change c as well). Determine the new value for $\dot{\theta}$, and call it $\dot{\theta}_1$. Now each $u_i = c_i \dot{\theta}_1$, so $c_i = u_i / \dot{\theta}_1$ and we have determined $C = u / \dot{\theta}_1$.

Code example:

```
// Calculate coupling matrix C:

// Assume "mobod" is a mobilized body whose 0th mobility has been scaled
// so that its generalized coordinate is the angle theta.
state.updU() = 0;
mobod.setOneU(state, MobilizerUIndex(0), 1); // thetadot_0 = 1
system.project(state, 1e-10,
    yWeights, cWeights, yErrEst, // dummies; see below
    System::ProjectOptions::VelocityOnly);
// Now calculate C.
const Vector C = state.getU() / mobod.getOneU(state, MobilizerUIndex(0)); // / thetadot_1
state.updU() = 0;

// Note: you can declare these dummies for the project() call above.
const Vector yWeights(state.getNY(), 1);
const Vector cWeights(state.getNMultipliers(), 1);
Vector yErrEst;
```

Mapping spatial forces to generalized forces

If you can collect the muscle's generated spatial body forces *F* into an array with an entry for each body (zero where the muscle does nothing), you can generate the equivalent generalized forces using Simbody's oddly-named method `calcInternalGradientFromSpatial()`.

Code example:

```
// Calculate the joint torques f equivalent to the muscle forces F, method 1:

Vector_<SpatialVec> F; // Indexed by MobilizedBodyIndex
muscle.setTension(s);
```

```
// ... obtain F from the muscle somehow
Vector f;
matter.calcInternalGradientFromSpatial(state, F, f);
```

An alternative is to let Simbody gather up all the body and generalized forces produced by all force elements, and work with the difference between inactive and activated muscle.

Code example:

```
// Calculate the joint torques f equivalent to the muscle forces F, method 2:

Vector f0, f1, f;
// Start with the muscle off
muscle.setTension(0);
system.realize(Stage::Dynamics);
matter.calcInternalGradientFromSpatial(state,
    system.getRigidBodyForces(state,Stage::Dynamics),
    f0);
f0 += system.getMobilityForces(state,Stage::Dynamics);

// Now turn on the muscle
muscle.setTension(s);
system.realize(Stage::Dynamics);
matter.calcInternalGradientFromSpatial(state,
    system.getRigidBodyForces(state,Stage::Dynamics),
    f1);
f1 += system.getMobilityForces(state,Stage::Dynamics);

// f is the change in generalized forces due to the muscle
f = f1 - f0;
```

Calculating the moment arm

With `C` and `f` calculated as above, the moment arm calculation is easy:

```
Real momentArm = ~C * f / s;
```

Calculating effective joint torque

Assuming mass properties are available, we can test consistency between moment arm and the resulting dynamics. We can apply a muscle tension s, calculate moment arm r_θ about some angular quantity θ as above, and also calculate generalized accelerations $\dot{u}$ and constraint multipliers λ using the equations of motion, that is, by calling realize(Acceleration) in Simbody. Then given $\dot{u}$ and λ we would like to determine the torque τ_θ that must have been applied across the composite joint that defines θ, and verify that $\tau_\theta = r_\theta s$. We'll refer to the following generic model in the discussion below:

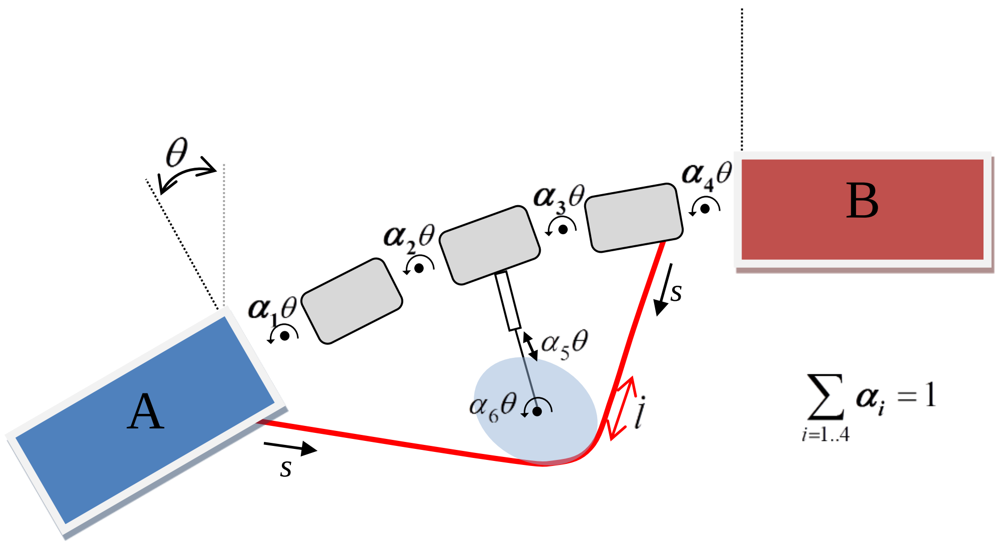

Figure 3: Idealized model incorporating most moment arm difficulties. The angle of interest θ relates the orientation of body B and body A, but is the sum of four coupled angles $\theta_{1\text{-}4}$ whose values are related to θ by the coupling constants $\alpha_{1\text{-}4}$. Other coupled coordinates $\theta_{5,6}$ are present and affect muscle length via a workless wrap surface, but are not part of the definition of θ. The muscle insertion point is not on either of the bodies used to define θ. See text for discussion.

Assume that θ is defined as

$$\theta \triangleq \sum \theta_k \tag{19}$$

That is, the angle of interest is determined by the sum of the individual angles at several joints. If we further assume that these angles are kinematically coupled to θ with instantaneous coupling constants α_k then*

$$\theta = \sum \alpha_k \theta \quad \Rightarrow \quad \sum \alpha_k = 1 \tag{20}$$

Note that these are not necessarily *all* the coupled coordinates in the system, just those whose angles determine θ. In Figure 3 only $\alpha_{1\text{-}4}$ (shown in bold) affect θ.

Now the power P generated by a muscle in our model is just the product of its tension and contraction velocity, that is

$$\begin{aligned} P = s\dot{l} &= r_\theta s \dot{\theta} \\ &\triangleq \tau_\theta \dot{\theta} \end{aligned} \tag{21}$$

* We're assuming without loss of generality that θ_k=0 when θ=0; any constant offset can be eliminated by shifting the reference point from which a θ_k is measured, and in any case the derivation involves time derivatives of these angles in which the constants drop out.

with the second equality following from Equation (11), and where we have now defined the effective torque $\tau_\theta \triangleq r_\theta s$. We would like to apply a muscle tension *s*, evaluate the system dynamics, then calculate τ_θ using inverse dynamics so that we can verify that it is consistent with the moment arm r_θ. However, inverse dynamics does not provide τ_θ directly. Instead, it gives us a torque τ_i at each of the joint degrees of freedom. Since all constraints are presumed non-working, the total power should be the sum of power contributions at the θ_k angles that determine θ, that is

$$P = \sum \tau_k \dot{\theta}_k = \sum \tau_k \alpha_k \dot{\theta} = \tau_\theta \dot{\theta} \tag{22}$$

$$\Rightarrow \quad \tau_\theta = \sum \alpha_k \tau_k \tag{23}$$

Equation (23) gives the algorithm we need to compute the effective torque τ_θ from the joint torques τ_k. There remains one unresolved issue: in a multibody model using generalized coordinates, inverse dynamics yields *generalized* forces, which are not necessarily torques.

Obtaining joint torques from generalized forces

Note that the angles θ_k and angular rates $\dot{\theta}_k$ are not necessarily the same as the corresponding generalized coordinates q_k and speeds u_k, since OpenSim allows arbitrary generalized coordinates. Similarly, the torques τ_k are not generalized forces f_k since those must be scaled to be consistent with the generalized speeds.

Denote the scaling of the k^{th} generalized speed w_k such that $\dot{\theta}_k = w_k u_k$. Then we must have $\tau_k = f_k / w_k$ because $f_k u_k$ must have physically meaningful units of power. So now we can use inverse dynamics to calculate $f = M\dot{u} + G^T \lambda$ and calculate $\tau_\theta = \sum (\alpha_k / w_k) f_k$ and compare it with $r_\theta s$.

However, from the definition of *C* in equation (13), and the definitions of w_k and $\dot{\theta}_k$ above, we see that $u_k = c_k \dot{\theta} = \dot{\theta}_k / w_k = \alpha_k \dot{\theta} / w_k \Rightarrow c_k = \alpha_k / w_k$. Thus we have can simplify our calculation of τ_θ to

$$\tau_\theta = \sum c_k f_k = C_\theta^T f \tag{24}$$

Where C_θ is just *C* with zeroes replacing rows that do not correspond to generalized coordinates that contribute to the definition of θ, and f is the complete set of generalized forces.

Note that using this method the generalized coordinate weights conveniently drop out of the calculations.

---

[1] From http://www.xuvn.com/chiapex/physiology/skeletal_muscle_structure.htm, accessed 24 Oct 2010.
[2] Storace, A., and Wolf, B. Functional analysis of the role of finger tendons. *J. Biomechanics* 12(8):575-578 (1979)
[3] Gonzalez, R., Buchanan, T., and Delp, S. How muscle architecture and moment arms affect wrist flexion-extension moments. *J. Biomechanics* 30(7):705-712 (1997)
[4] Seth, A., Sherman, M., Eastman, P., Delp, S. Minimal formulation of joint motion for biomechanisms. *Nonlinear Dynamics* 62:291-303 (2010)